TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: N2612 – Elektrotechnika a informatika
Studijní obor: 1802T007 – Informační technologie

Zpracování videosignálu v obvodu FPGA

Video signal processing on a FPGA circuit

Diplomová práce

Autor: **Bc. Jan Tůma**
Vedoucí práce: Ing. Martin Rozkovec, Ph.D.

V Liberci 2. 1. 2013

Originál zadání

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

2. 1. 2013

Jan Tůma

Abstrakt

V rámci diplomové práce jsou nejprve s využitím programovacího jazyka VHDL navrženy a odzkoušeny vybrané algoritmy úpravy videosignálů pro aplikaci pomocí hradlových polí FPGA. Pro vlastní realizaci pak bylo využito vývojové desky Digilent Atlys , která je osazena obvodem Spartan-6 XC6SLX45. Celá úloha je řešena jako systém, který je možné připojit mezi zdroj video signálu a zobrazovací zařízení. Tímto zařízením je možné pro zdrojový video signál provést následující úpravy: vytvoření černobílého obrazu, gama korekci, prahování, vytvoření negativu, aplikace konvolučního filtru, detekce hran. Vzhledem k paralelním výpočtům pomocí hradlových polí tak bylo dosaženo zpracování video signálu v reálném čase.

Klíčová slova: Hradlová pole, video signál, úprava barev, konvoluční filtr, programovací jazyk VHDL.

Abstract

First part of the diploma thesis is focused on video signal processing algorithms suitable for implementation based on field-programmable gate array (FPGA) devices. Selected algorithms were programmed using the VHDL language and tested. In the second part of the thesis, these algorithms were implemented using the Digilent Atlys development board equipped with Spartan-6 XC6SLX45 device. This system can be connected between a video signal source and video rendering device using standard video cables. The following video processing operations can be performed on the input stream of video data: grayscale image, gamma correction, thresholding, negative image, convolution filtering and edge detection. Because of parallel computation implemented in FPGA devices real-time video signal processing has been achieved.

Keywords: Field-programmable gate array, video signal, color adjustment, convolution filter, VHDL programming language

Obsah

Seznam obrázků

Seznam použitých zkratek

CLB	configurable logic block
CPLD	complex programmable logic device
CRT	cathode ray tube
DVI	digital visual interface
EDK	embedded development kit
FPGA	field-programmable gate array
HDL	hardware description language
HDMI	high-definition multimedia interface
IOB	input/output block
JTAG	joint test action group
NTSC	national television system committee
PAL	phase alternating line
PSM	programmable switch matrix
SDK	software development kit
SECAM	séquentiel couleur à mémoire
TMDS	transition-minimized differential signaling
USB	universal serial bus
VHDL	very-high-speed integrated circuit hardware description language
XPS	xilinx platform studio

Úvod

Téma diplomové práce spadá do oblasti multimediálních technologií, což je oblast, která se dnes velice rychle rozvíjí a nabývá stále většího významu snad ve všech sférách lidské společnosti. Vybavení nejmodernějšími technologiemi je vzhledem k jejich nárokům na technické vybavení a tedy většinou i vyšší ceně asi stále doménou velkých společností, nicméně tempo miniaturizace a velkosériová výroba umožňuje velice rychle dostat tyto technicky vyspělé prostředky až do jednotlivých domácností, tedy k běžným spotřebitelům. Příkladem mohou být dnes již cenově poměrně dostupná různá digitální záznamová zařízení jako např. videokamery a digitální fotoaparáty umožňující video záznam ve stále vyšším rozlišení.

Cílem mé práce je nepatrně přispět ke zpřístupnění zpracování videosignálu i v podmínkách, kdy není efektivní si pořizovat drahou a mnohdy zbytečně robustní profesionální techniku, ale za relativně dostupnou cenu získat poměrně rychlý prostředek ke zpracování základních video efektů. Takový přístroj může pak získat uplatnění např. v domácím prostředí jako poloprofesionální střižna. Hlavním cílem však bylo ověřit možnosti aplikace FPGA obvodů pro zpracování videosignálů. Tedy umožnit zhotovení takového přístroje, který by byl (na rozdíl od dosud běžně používaných prostředků pro dávkové zpracování videa a jeho následného přehrávání) umístěn mezi zdroj signálu (DVD přehrávač, počítač) a zobrazovací zařízení (televizor, monitor) a v podstatě jen s velmi malým zpožděním, řádově milisekundy, by upravený zdroj videosignálu "předal" k uživateli ke zhlédnutí.

Text diplomové práce je rozdělen do několika kapitol, první dvě kapitoly jsou koncipovány jako stručná rešerše relevantních oblastí k tématu práce. Další kapitola popisuje zvolené hardwarové a softwarové prostředky a vlastní řešení diplomové práce je popsáno v posledních dvou kapitolách. V kapitole č.4 jsou pomocí blokových schémat popsána navržená řešení zvolených úprav video signálů. Vzhledem k dostupnému technickému vybavení, které bylo omezené počtem hradlových polí (tj. výpočetní kapacitou), byly algoritmy rozděleny do dvou skupin - úprava barev a úpravy pomocí konvolučních filtrů. Každá z těchto skupin pak byla v FPGA obvodu realizována samostatně. Kapitola č.5 popisuje vlastní realizaci zpracování videosignálu na konkrétním příkladě.

1 Diskrétní realizace a přenos obrazových dat

Zpracování obrazových dat je otázkou nejen příslušných algoritmů pro jejich zpracování, ale i otázkou digitalizace spojitého obrazu, zpracování a uložení diskrétních hodnot a v neposlední řadě i otázkou přenosu těchto dat [3], [7], [9]. Tato kapitola je koncipována jako stručný úvod k teoretické části přístupu a využití metod pro diskrétní realizaci a přenos obrazových dat.

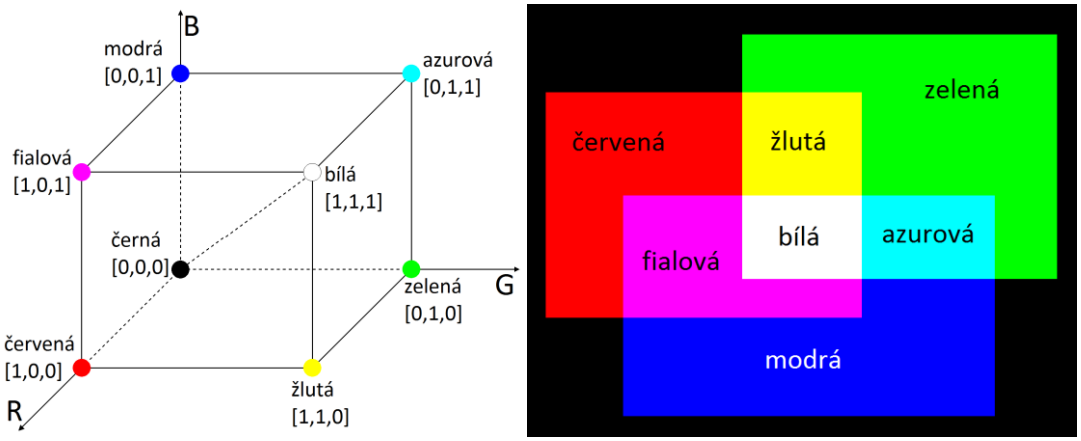
1.1 Reprezentace obrazových dat v číslicových systémech

Pro potřeby digitálního zpracování obecně spojitého signálu (obrazu) je potřeba nejprve data navzorkovat do konečného počtu snímků (diskretizace v čase) a bodů (diskretizace v prostoru). Pro dvojrozměrný obraz jednoho snímku se hodnota jasu jednotlivých bodů (pixelů) může vyjádřit pomoci funkce Y = $f(x, y)$, kde dvojice hodnot x a y je libovolnou kombinací z konečné množiny hodnot na intervalu $<x_{min}, x_{max}>$, resp. $<y_{min}, y_{max}>$ a odpovídají souřadnicím pixelů v daném snímku. Takto zavedená hodnota jasu by však popisovala vlastnosti pouze monochromatického obrazu. Pro barevný obraz je třeba rozšířit popis i o další hodnoty obrazového bodu podle modelu barevného prostoru. Např. při využití tzv. RGB prostoru (viz dále), by popis jasu jednoho pixelu byl dán funkčními hodnotami $f_R(x, y)$, $f_G(x, y)$, $f_B(x, y)$, tedy postupně pro jednotlivé barevné složky R, G, B.

Volba barevného prostoru je tedy dalším krokem pro vyjádření informace o vlastnostech daného obrazového bodu a je v podstatě do značné míry závislá na prostředí, resp. technice, která bude použita pro interpretaci digitálního obrazu. Dále je popsáno několik často používaných modelů:

RGB model - je založen na aditivním způsobu kombinace barev červené (Red), zelené (Green) a modré (Blue). Vlnové délky těchto tří barev se skládají (sčítají) a vytváří se tak výsledné barevné spektrum, které je vnímáno lidským okem (nebo

zachyceno fotoaparátem či jiným světelným senzorem). Tento model je standardně používán např. u monitorů a datových projektorů. Na obr. 1.1.a. je znázorněna prostorová interpretace, kde jednotlivé barevné složky tvoří osy trojrozměrného kartézského prostoru a libovolný bod uvnitř a na obvodu vyznačené jednotkové krychle pak představuje jedinečnou hodnotu výsledné barvy. Spojnice bodů (0,0,0) a (1,1,1) je množinou bodů šedotónového obrazu. Příklad skládání barev je uveden na obr. 1.1.b.

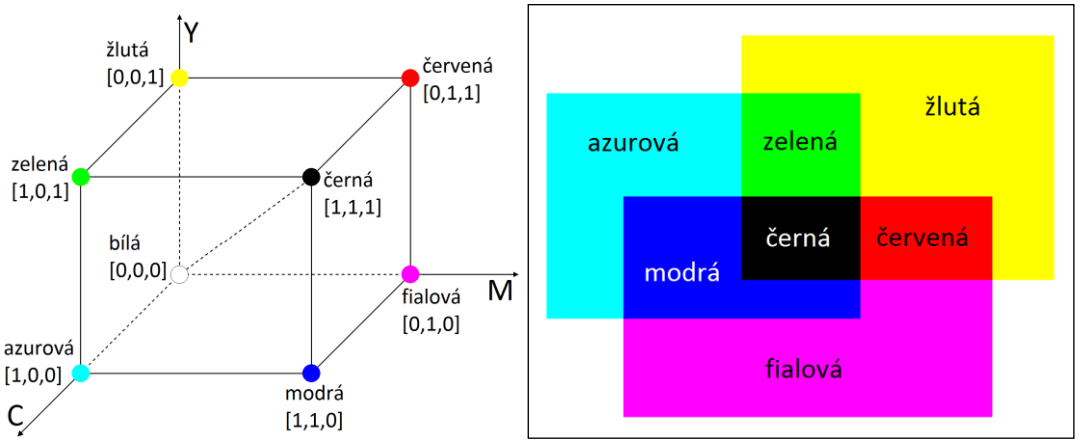


Obr. 1.1. Zobrazení modelu RGB a)v trojrozměrném prostoru, b)skládáním barev

Použijeme-li k uložení hodnoty každé složky modelu RGB např. 1 byte, tj. hodnoty v intervalu <0, 255>, je celkový počet kombinací barevného zobrazení tohoto modelu dán mocninou 2^{24} (tj. 16 777 216 barev). Hodnotě 24 (mocnině 2) se také říká bitová hloubka obrazu.

CMYK model - je dalším barevným modelem, který však místo aditivního skládání barev používá subtraktivní míchání tří barev: azurové (Cyan), purpurové (Magenta) a žluté (Yellow). Tento model se používá při tisku. Lidské oko totiž nevnímá nanesený pigment na papíře, ale barevné spektrum, které vznikne jako část bílého světla při odrazu od tohoto pigmentu (část barevného spektra bílého světla je tedy pohlcena - odčítá se). Písmeno K vyjadřuje doplnění uvedených tří barev o složku černou (blacK), která se z praktického hlediska používá místo subtraktivního skládání černé barvy

z uvedených tří komponent, viz obr. 1.2.a. V reálných podmínkách se totiž z těchto komponent většinou dosáhne něco jako špinavě hnědá nebo šedá barva. Na obr. 1.2. je opět uveden prostorový model se znázorněním základní barevné palety.



Obr. 1.2. Zobrazení modelu CMYK a)v trojrozměrném prostoru, b)skládáním barev

V praxi jsou používány i další barevné modely, které většinou vznikly na základě specifických požadavků. Např. s rozvojem šíření barevného televizního signálu byl zaveden model YUV, který umožnil kompatibilní přenos černobílého a barevného signálu. Hodnota Y zde neoznačuje žlutou barvu, ale velikost jasu daného bodu, ve složkách U a V je pak "redukována" informace o barvě. Primárně byl tento model používán pro televizní normu PAL, obdobně byl navržen model YC_BC_R pro normu SECAM a model YIQ pro americkou televizní normu NTSC. Za zmínku stojí i barevné modely HSV a HLS vyvinuté pro potřeby grafiků, které jsou navrženy na principu barevného tónu a sytosti spolu s jasovou hodnotou, resp. světlostí. Jejich barevný prostor není již dán krychlemi, ale dvojicí jehlanů. Praktické využití těchto dvou modelů je např. při tvorbě vzorníků, kde je vhodné definovat vlastnosti barvy pomocí pojmů jako je sytost, světlost a dominantní barva.

Z pohledu zpracování obrazových dat je ještě vhodné doplnit informaci o možnostech uložení dat z hlediska jejich "umístění" v obraze. Rozlišuje se v podstatě forma rastrové a vektorové grafiky. Zatímco rastrová (bitmapová) grafika umožňuje aplikaci poměrně jednoduchých algoritmů pro další zpracování obrazu právě k dosažení

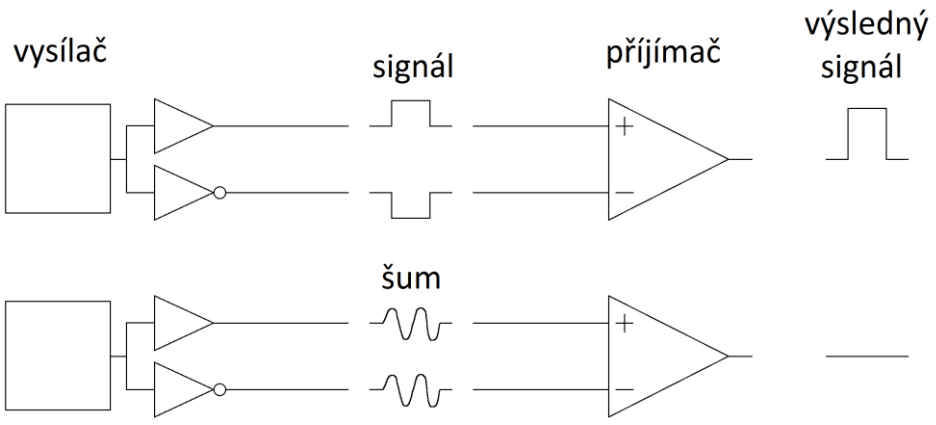
různých video efektů, ale je náročná na velikost paměti a změnu měřítka, tak vektorová grafika uchovává informaci o tvarech objektů a je vhodná především pro potřebu změn velikosti a tvaru obrazu a menších nároků na paměť počítače.

1.2 Přenos obrazových dat

V této části jsou popsány tři protokoly pro přenos diskrétních dat, přičemž větší prostor je dán protokolu TMDS (Transition Minimized Differential Signaling), který jsem použil při řešení praktické části.

TMDS protokol je dnes již standardem, který se hlavně využívá pro digitální přenos nekomprimovaného videosignálu mezi zdrojem videa a přijímačem. TMDS specifikuje elektrickou vrstvu, protokol a formát dat a byl vyvinut společně s DVI 1.0 (Digital Visual Interface) [1].

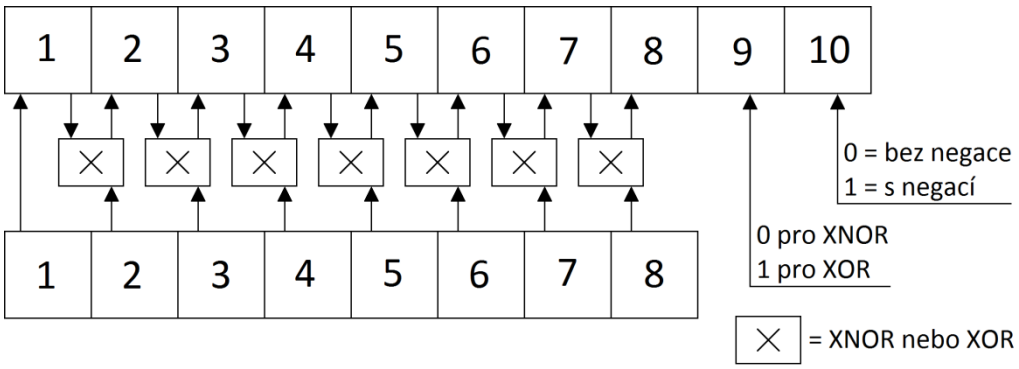
TMDS, jak již z názvu napovídá, využívá pro přenos dat rozdílové vedení signálů. Pro přenos každého signálu jsou použity dva vodiče, kterými je přenášena záměrně hodnota požadovaného signálu s opačnými znaménky, což neplatí o případném šumu. Princip kompenzace šumu je znázorněn na obr. 1.3, kde je zvlášť uveden princip přenosu pouze užitečného signálu, který je na přijímači zachován, a zvlášť je znázorněn přenos případného šumu, který je naopak přijímačem odstraněn. Díky tomu je zajištěna větší odolnost proti rušení při velkých rychlostech přenosu dat a na delší vzdálenosti.



Obr. 1.3. Schéma principu rozdílové signalizace

Přenos obrazových dat ve formátu RGB probíhá současně pro každou barvu, tj. pro každou barevnou složku je vyhrazen jeden datový kanál. Vedle datových kanálů je specifikován ještě kanál pro přenos hodinového signálu. Maximální přenosová frekvence, jakou je možné přenášet pixely je 165MHz. V případě potřeby na vyšší datové toky je dle specifikace TMDS možné použít další trojici datových kanálů. Pro potřeby této diplomové práce ovšem stačila přenosová kapacita jedné trojice datových kanálů.

Dalším specifikem TMDS je snaha o minimalizaci přechodových stavů, k tomu se využívá speciálního kódování, kdy se přenášených 8bitů upraví a doplní o další dva bity, tedy celková informace se zakóduje na 10 bitů. Při kódování se využívá funkce XOR nebo XNOR a to v závislosti na počtu „jedniček“ v kódovaném slově. V případě, že je tento počet roven hodnotě 4 nebo větší a první bit slova je roven 0, tak se použije funkce XNOR, v ostatních případech se použije funkce XOR. Tímto postupem se docílí snížení počtu přechodů 0-1 a 1-0 v přenášeném slově, přičemž do devátého bitu se pak uloží informace o tom, jestli bylo použito XOR nebo XNOR. Následně se ještě na základě hodnot předchozího bytu provede případná negace aktuálně kódovaného bytu pro eventuelní vyrovnání úrovně stejnosměrného proudu. Informace o provedení této negace je uložena v posledním desátém bitu kódovaného slova. Tento proces kódování dat je schematicky znázorněn na následujícím obr. 1.4.



Obr. 1.4. Kódování dat v TDMS protokolu

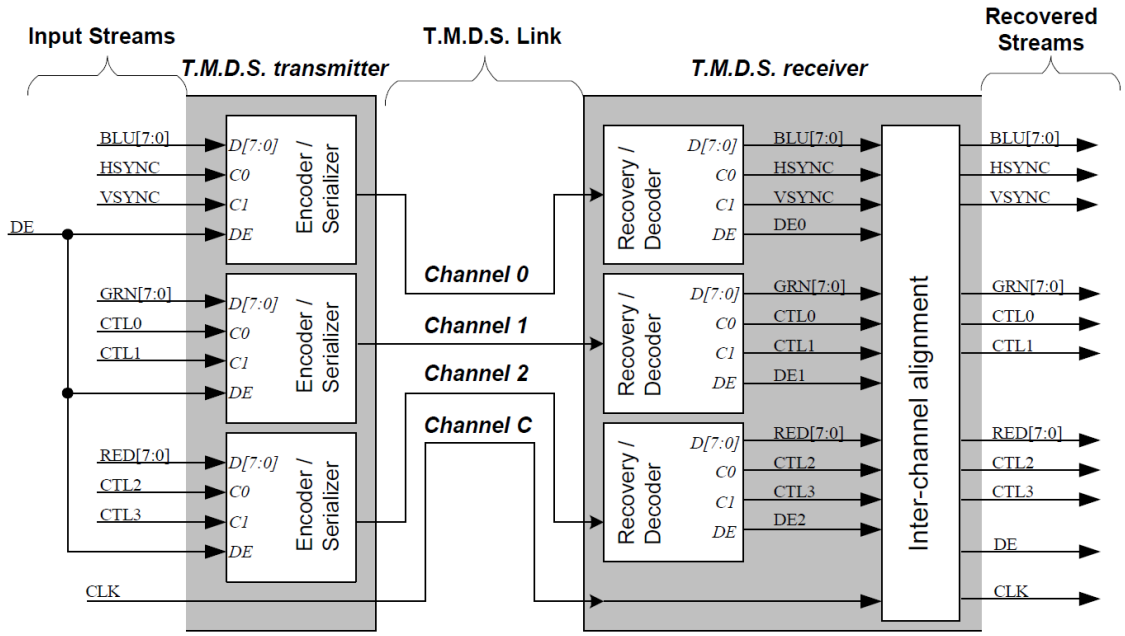
Pro vysílání signálu používá TMDS vysílač tři identické kódovací jednotky (viz obr. 1.5.), pro každý RGB kanál jednu, v kterých zároveň dochází i k serializaci dat. Vstupem každé této jednotky jsou dva řídící signály, osm bitů obrazových dat (jedna barva) a signál DE (Data Enable). V závislosti na hodnotě signálu DE jednotka zakóduje a vysílá buď obrazová data nebo řídící signály.

Řídící signály se vysílají v době, kdy není vysílán viditelný obraz. Jedná se o signály obsahující informaci o horizontální (HSYNC) a vertikální (VSYNC) synchronizaci a dále signály CTL0 až CTL3, které musí být v logické hodnotě 0. V následující Tabulce 1. je uvedeno kódování výstupních slov z jednotlivých jednotek vysílače TMDS na základě hodnot C0 a C1, jejichž význam je patrný z obr.1.5.

Tabulka 1. Kódování řídících bitů

Vstupní řídící bity		**Výstupní zakódované slovo**
C0	**C1**	
0	0	0010101011
0	1	0010101010
1	0	1101010100
1	1	1101010101

TMDS přijímač zajišťuje příjem sériových dat, jejich dekódování a deserializaci, k čemuž slouží tři identické dekódovací jednotky po jedné pro každý kanál (viz obr. 1.5.). Při prvním vysílání dat ještě musí dojít k inicializační synchronizaci mezi zobrazovacím zařízením a zdrojem dat na všech kanálech, čehož se dosáhne na základě přijímaných řídících dat. Až posléze může přijímač tato synchronizovaná dekódovaná data předávat k dalšímu zpracování. Opět tak získáme nezměněná RGB data a řídící data.



Obr. 1.5. Schéma přenosu kódovaných dat u TMDS protokolu [1]

První rozhraní využívající TMDS protokol bylo DVI, dnes je ovšem pro tento protokol nejrozšířenějším rozhraním HDMI (High-Definition Multimedia Interface), které se stalo standardem v multimediálních zařízeních, včetně osobních počítačů.

Rozhraní DVI slouží primárně k přenosu nekomprimovaných obrazových dat mezi osobními počítači a jejich zobrazovacími jednotkami, k čemuž využívá právě výše uvedený protokol TMDS. Jedná se o rozhraní, podporující technologii „plug and play“. Rozhraní DVI bylo vytvořeno pro přenos obrazových dat v digitální podobě, umožňuje ovšem i přenos analogového obrazuje a zajišťuje tak zpětnou kompatibilitu s analogovými systémy. Využívá 29 pinového konektoru, ve kterém nemusí být všechny piny aktivně využity. Podle toho, jaké piny jsou zapojeny, se dále rozlišují tři verze konektoru:

DVI-A	–	pouze analogové piny, nepodporuje přenos digitálních dat
DVI-I	–	kombinuje analogové i digitální piny
DVI-D	–	pouze digitální piny

DVI-I a DVI-D se můžou ještě vyskytovat ve dvou modifikacích a to single link (3 páry TMDS kanálu pro přenos obrazových dat) nebo ve verzi dual link (6 páru).

Rozhraní HDMI bylo vytvořeno v roce 2002 a slouží k přenosu nekomprimovaných digitálních audiovizuálních signálů a dalších řídicích dat pro speciální funkce, jako je např. ovládání více zařízení jedním ovladačem. Stejně jako rozhraní DVI podporuje technologii „plug and play“. Hlavně ale umožňuje na rozdíl od DVI přenášet jak obrazová, tak i vícekanálová audio data, a to ve vysoké kvalitě. Je kompatibilní s rozhraním DVI, což ale neplatí opačně právě vzhledem k tomu, že DVI neumí přenášet zvuková data. Ačkoliv se jedná o poměrně nové zařízení, velice rychle se rozšířilo do veškeré multimediální spotřební elektroniky.

2 Video efekty

Jak jsem již uvedl v předchozí kapitole, téma zpracování obrazového signálu je velmi obsáhlé a jeho vývoj je s neustálým rozvojem informačních technologií v podstatě stále otevřený. V digitální podobě lze signál v podstatě jakkoliv upravit, měnit intenzitu, jas i rozložení barev, tím můžeme pak nejrůznějšími způsoby ovlivnit i vlastní topologii obrazu, tj. některé objekty zvýraznit, jiné potlačit či přesunout jinam, včetně přidání prvků nových. S tím souvisí také i různé retuše, barevné kompozice ap.

V této kapitole se proto zaměřím na stručné přehledové zpracování často používaných video efektů [4], [9], zejména pak těch, které budou obsahem mé praktické části diplomové práce.

2.1 Úprava barev

Požadavek na úpravu barev v obraze je velmi častý a z různých důvodů. Pomocí úprav barev můžeme vylepšit celkový vjem např. při příliš přesvícených nebo tmavých částí, opravit a zkorigovat barvy a tónový rozsah, opravit nežádoucí vliv použité techniky, upravit obraz s ohledem na technologii barevného tisku ap. V této kapitole uvedu jen některé úpravy, které pak budou aplikovány v praktické části diplomové práce.

2.1.1 Šedotónový obraz

V barevném prostoru RGB (viz kapitola 1.1.) se šedotónový obraz, tj. rozlišení intenzity bodu obrazu pomocí stupně šedi, vytvoří v případě, že všechny barevné složky mají stejnou hodnotu, tedy pro každý pixel platí R = G = B. Z hlediska prostorového vyjádření dle obr. 1.1.a. je množinou takových bodů tělesová úhlopříčka mezi body (0, 0, 0) a (1, 1, 1). Pokud bychom měli z hlediska zpracování barevného obrazu za úkol vytvořit obraz šedotónový, znamenalo by to nějakou relací vyjádřit libovolný bod

jednotkové krychle (3D prostor) jako umístění bodu v rámci úsečky (1D prostor). Takovou jednoduchou relací je např. prostý průměr z jednotlivých barev, tj. Y = (R + G + B)/3. Hodnota Y je pak hodnota jasu šedotónového obrazu vyjádřená v rozsahu hodnot <0, 1> a pro vyjádření v RGB prostoru pro daný bod platí R = G = B = Y. Tím získáme převod barevného obrazu na obraz pouze s jasovou hodnotou. Jelikož lidské oko nevnímá všechny barvy stejně, používá se pro lepší výsledek místo obyčejného průměrování empiricky definovaného váženého průměru.

$$Y = 0.299\,R + 0.587\,G + 0.114\,B$$

2.1.2 Gama korekce

Korekce mezi požadovaným a zobrazovaným jasem byla požadavkem zejména u analogových CRT monitorů, nicméně její uplatnění je možné využít i v rámci současných zobrazovacích technik. Problém spočívá v tom, že technické řešení monitorů není schopné přesně interpretovat požadovaný jas, ale zobrazí jej s určitým nelineárním zkreslením, které je dáno vztahem

$$i_1 = i_0^{\gamma}, \qquad (1)$$

kde i_0 je vstupní hodnota intenzity daného pixelu do zobrazovacího zařízení a i_1 je intenzita skutečně zobrazená. Gama koeficient v mocnině je dán konkrétním přístrojem a může se mezi různými přístroji lišit. Výrobce jej také většinou ve svých technických datech uvádí a u již zmíněných CRT monitorů se pohybovala obvykle v rozmezí hodnot 0.22 až 0.28. Vlastní gama korekce pak není nic jiného, než-li vyjádření požadované intenzity i_0 z rovnice (1), tedy odstranění nelinearity pomocí inverzní funkce. Pokud tedy pro konkrétní intenzitu i daného bodu provedeme nejprve gama korekci podle vztahu:

$$i_0 = i^{\frac{1}{\gamma}}, \qquad (2)$$

a takto korigovanou hodnotu i_0 pak dosadíme do vztahu (1) (tj. po zpracování daného zobrazovacího zařízení) bude výsledná intenzita i_1 rovna požadované i.

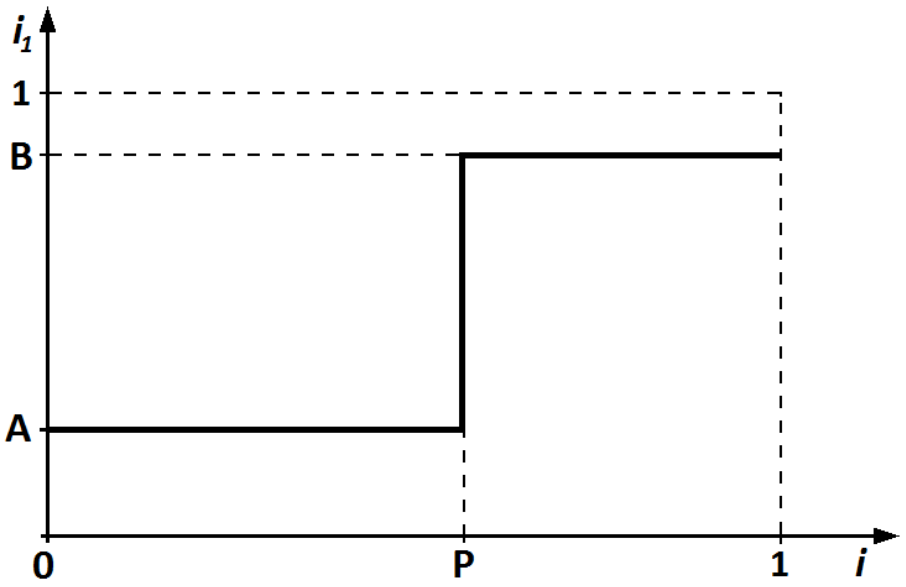
Důležitou vlastností vztahu (1), resp. (2) je, že pokud budeme uvádět rozsah možných hodnot u intenzity v rozmezí <0, 1>, tak pro krajní hodnoty (tedy bílou a černou barvu) se hodnota intenzity nemění. Z toho plyne, že gama korekcí se nemění nejsvětlejší a nejtmavší body obrazu, ale mění se rozložení jasů uvnitř rozsahu. Tím vzniká efekt zesvětlení resp. ztmavení části obrazu, čehož lze záměrně využít i pro jiné efekty, než-li jen odstranění uvedených nelinearit zobrazovacích přístrojů.

2.1.3 Prahování

Úprava obrazu prahováním vychází z funkce, která obecně upravuje hodnoty vstupu podle následujícího vztahu

$$f(x) = \begin{cases} A\ pokud\ x \leq P \\ B\ pokud\ x > P \end{cases} \qquad (3)$$

kde P je hodnota "prahu" z oboru definičních hodnot nezávisle proměnné *x*. Z pohledu zpracování obrazu můžeme za nezávisle proměnnou použít původní intenzitu obrazového bodu *i* a výstupem rovnice (3) pak bude hodnota intenzity i_1 upraveného bodu. Tato relace znázorněna na obr. 2.1.



Obr. 2.1. Grafické znázornění funkce prahování pro jednu úroveň

Důležitá je volba prahové hodnoty, která může podstatně ovlivnit kvalitu celého výsledku, v extrémním případě může být výsledný obraz tvořen z bodů jediné barvy. Jako vhodná hodnota prahu se může použít např. medián, nebo střední hodnota. Pokud např. zvolíme hodnoty A = 0 a B = 1, bude výsledkem černobílý obraz (složený pouze z bílých a černých bodů), což při vhodně určené hodnotě prahu lze použít třeba pro transformaci barevného obrazu na binární, kde body náležející zobrazenému objektu budou mít hodnotu 1 a body pro pozadí hodnotu 0.

Speciálním případem může být i tzv. ohraničené prahování. Příklad je uveden pomocí 3 prahových hodnot D, T, U a dvou výstupních úrovní Y a V. Výsledná intenzita i_1 takto zadaného příkladu je dána vztahem (4).

$$i_1 = \begin{cases} i & pro \;\; 0 \leq i < D \\ Y & pro \;\; D \leq i < T \\ V & pro \;\; T \leq i < U \\ i & pro \;\; U \leq i \leq 1 \end{cases} \tag{4}$$

2.1.4 Negativ

Úprava obrázku na tvar označovaný jako negativ se používá např. pro skenování negativních předloh, před skládáním obrazů nebo také v souvislosti s funkcemi typu eroze, dilatace apod. Výstupní intenzita obrazového bodu negativního obrazu i_1 se vypočítá z původní intenzity i podle jednoduchého vztahu:

$$i_1 = 1 - i\,. \tag{5}$$

2.2 Filtrace obrazu

Použití filtrů při úpravě obrazu má mnoho aplikačních využití. Filtrace může ovlivnit zrnitost obrazu, zvýraznit nebo potlačit přechodová místa obrazu (detekce hran). V podstatě se jedná o změnu intenzity obrazového bodu s využitím informace z jeho okolí. Podle způsobu vyhodnocení těchto informací se filtry dále rozlišují na lineární a nelineární.

2.2.1 Lineární filtry

U lineárních filtrů se využívá změny intenzity obrazového bodu na základě informace z jeho okolí. K tomu se používá tzv. konvolučních filtrů, které pomocí konvoluce upraví vlastnosti daného bodu na principu váženého součtu vlastností okolních bodů. Příslušné váhové koeficienty jsou obsaženy v konvoluční matici (označované také jako jádro, nebo anglickým slovem kernel), jejíž rozměr je z praktického hlediska většinou čtvercový s lichým počtem řádků, resp. sloupců. Tím je totiž jednoznačně určen středový prvek této matice, který je pak přiřazen změně konkrétního bodu obrazu. Pro dvourozměrný diskrétní obraz můžeme výpočet konvolučního filtru zapsat ve tvaru:

$$f(x,y) * g(x,y) = \sum_{i=-k}^{k} \sum_{i=-k}^{k} f(x-i, y-i) \cdot g(i,j) \qquad (6)$$

kde $f(x, y)$ jsou obrazové body předlohy

$g(x, y)$ jsou koeficienty konvoluční matice

k je parametr určující velikost konvoluční matice - $(2k+1)$ řádků

Rovnice (6) popisuje výpočet jednoho bodu nového obrazu. Celý nový obraz se pak vytvoří postupnou změnou souřadnic x a y v jejich definovaném rozsahu. Prakticky to znamená, že pro každou kombinaci diskrétních hodnot (x, y) bude vyhodnocena rovnice (6) jako součet součinů příslušných hodnot matice f a matice g, kdy matici g "soustředně" přiložíme na matici f v bodě (x, y).

Pro konkrétní aplikace zbývá určit rozměr konvoluční matice a její prvky. Větší rozměr matice dává sice možnost většího prostoru pro kvalitu změny obrazu - filtraci, zároveň se tím ale zvyšuje požadavek na počet operací, což může být zejména při snaze o rychlé zpracování obrazu nepříjemná komplikace. Proto se většinou používají matice o rozměru 3x3 nebo 5x5 prvků. Důležitá je i volba konkrétních hodnot prvků matice, tou lze totiž dosáhnout dalších specifických vlastností upraveného obrazu. Podle sestavení hodnot prvků konvoluční matice můžeme např. dále rozlišovat lineární filtry pro

- detekci hran,
- vyhlazování obrazu,
- ostření obrazu.

Filtry pro vyhlazování obrazu, označované také jako filtry typu dolní propust, jsou určeny zejména pro odstranění vyšších frekvencí intenzit v originálním obraze, tím se sice potlačí hlavně nežádoucí šum, ale také tím dojde k odstranění drobných obrazových detailů, čehož lze ovšem zase s výhodou využít při odstranění nežádoucí zrnitosti v obraze.

V literatuře (např. [9]) jsou často publikovány následující příklady pro matice rozměru 3x3:

$$\begin{bmatrix} 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 0 \end{bmatrix} \qquad \begin{bmatrix} 0 & \frac{1}{3} & 0 \\ 0 & \frac{1}{3} & 0 \\ 0 & \frac{1}{3} & 0 \end{bmatrix} \qquad \begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}$$

1D - horizontální 1D - vertikální 2D

Tyto matice představují filtraci pomocí průměrování postupně v horizontálním a vertikálním směru a v celém okolí daného bodu obrazu. Charakteristickou vlastností těchto filtrů je, že součet prvků v konvoluční matici je roven 1, což zajistí požadavek na zachování jasu v obraze, pokud by byla hodnota součtu vyšší jak 1, jas by byl vyšší a naopak.

Filtry pro ostření obrazu, označované jako filtry typu horní propust, jsou určeny pro ty případy, kdy je oproti filtrům typu dolní propust naopak žádoucí detaily v obraze zvýraznit. Nepříjemné ale je, že tím může dojít i ke zvýraznění nežádoucího šumu. Pro tyto filtry je charakteristické, že součet prvků konvoluční matice je roven 0. Pro názornost uvádím jeden opět často publikovaný příklad konvoluční matice typu horní propust:

$$\begin{bmatrix} -\frac{1}{9} & -\frac{1}{9} & -\frac{1}{9} \\ -\frac{1}{9} & \frac{8}{9} & -\frac{1}{9} \\ -\frac{1}{9} & -\frac{1}{9} & -\frac{1}{9} \end{bmatrix}$$

A příklad hodnot konvoluční matice pro ostření obrazu:

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

2.2.2 Nelineární filtry

Nelineární filtry nevyužívají pro novou hodnotu daného bodu v obraze průměrování z okolních hodnot, ale použijí z okolí bodu jen jednu vhodnou hodnotu, a tou pak nahradí hodnotu upravovaného bodu. Výhodou proti lineárním filtrům tedy je, že nepřidávají do nového obrazu další nové vlastnosti bodů. Podle principu výběru vhodné okolní hodnoty se rozlišuje:

- filtr minimum
- filtr maximum
- filtr medián

Filtr minimum, označovaný jako eroze, vybere z okolí daného bodu bod s minimální hodnotou intenzity, kterou pak dosadí místo původní hodnoty upravovaného bodu. Filtr minimum umožňuje zúžení tmavých oblastí na světlém pozadí a tím i rozšíření světlých oblastí na tmavém pozadí. Může se tedy použít pro potlačení šumu ve světlejších částech obrazu, nebo např. pro zeslabení čar na výkresech.

Filtr maximum, označovaný jako dilatace, vybere z okolí daného bodu bod s maximální hodnotou intenzity, kterou dosadí místo původní hodnoty upravovaného bodu. Filtr maximum umožňuje rozšíření tmavých oblastí na světlém pozadí a tím i zúžení světlých oblastí na tmavém pozadí. Má tedy opačný efekt, než filtr minimum a může se tak použít pro potlačení šumu v tmavších částech obrazu, nebo pro zesílení čar.

Filtr medián je založen na principu, kdy okolní hodnoty upravovaného bodu jsou seřazeny vzestupně a nová hodnota je vybrána jako prostřední hodnota této posloupnosti. Použití tohoto filtru je velmi účinné pro potlačení šumu a může se s výhodou využívat např. při vyhlazování zrnitosti v obrazech. Určitou nevýhodou však je, že zaobluje hrany objektů, což má za následek určité zkreslení původních tvarů.

2.3 Klíčování a mixování

Úpravy obrazu pomocí klíčování a mixování umožňují skládat do jednoho obrazu dvě (nebo více) předloh. Princip klíčování je založen na rozlišení jedné části obrazu od druhé, přičemž klíčem je výskyt obrazových bodů (pixelů) stejné nebo velmi podobné barvy - tyto pixely tvoří jednu část obrazu, druhou část tvoří všechny zbývající pixely. Důležité je, aby tato druhá část obrazu pokud možno neobsahovala barvu klíče. Přestože volba klíče může být podle potřeby libovolná, z praktického hlediska (daného i požadavky natáčení filmových scén v ateliérech) je nejvíce používanou barvou modrá nebo zelená (bluescreen nebo greenscreen). Při natáčení scén pak bylo pozadí (plátno, stěna) natřeno příslušnou barvou, která pokud možno nebyla v natáčené scéně. Vlastní klíčování pak tvořil jednoduchý proces, kdy se dva snímky "položily přes sebe", přičemž spodní tvořil požadované pozadí a horní natočenou scénu s klíčem pozadí. V horním snímku se pak body s klíčem označily za průhledné a vznikl tak dojem, že natočená scéna byla pořízena na pozadí spodního snímku. Tento postup lze aplikovat i na jakékoliv dva obrazy, kdy je jen technickou otázkou, jak se označí oblast s barvou klíče, problém není použít současně i více různých barevných klíčů. To pak umožní kombinovat různé obrazy (videosekvence, klipy, snímky) do jednoho tak, že ve výsledném obraze vidíme z každého jen tu část, kterou chceme. Jednoduchou modifikací této techniky je pak označení klíčovaných (barevně odlišených) pixelů za částečně průhledné, což umožní efekt prolínání dvou obrazů (scén) do sebe.

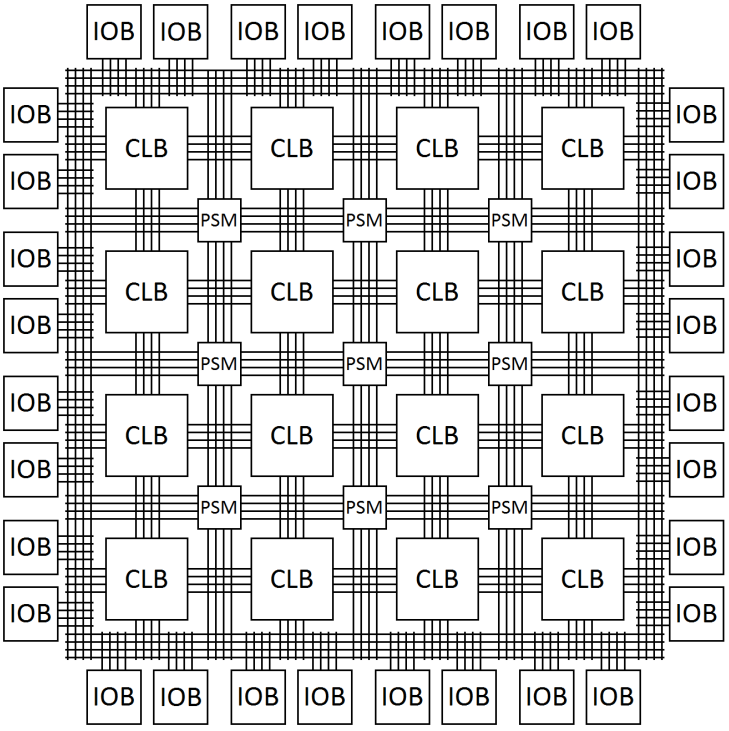
Na obdobném principu je založena i technika mixování dvou obrazů, přičemž místo klíče (barvy) se přímo definuje oblast, která bude v jednom z obrazů nahrazena motivem druhého obrazu. Klasickým příkladem této techniky je např. používání efektu obraz v obraze u televizorů (PIP - picture in picture). S příklady klíčování se můžeme setkat zase např. při televizních předpovědích počasí, kdy moderátor ve studiu je natáčen před barevnou scénou, která je pak nahrazena vlastní předpovědní mapou, nebo při natáčení nejrůznějších filmových scén, které by byly jinak nebezpečné, nebo i zcela nereálné (vznášení v prostoru ap.).

3 Prostředky k realizaci video efektů pomocí FPGA

Tato kapitola již popisuje vlastní část řešení diplomové práce. Jsou zde popsány hardwarové a softwarové prostředky, které jsem použil při návrhu algoritmů video efektů a jejich implementaci na FPGA obvod.

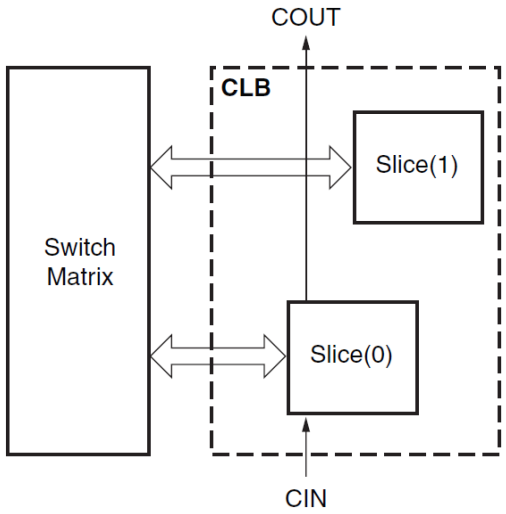
3.1 FPGA obvod Spartan 6

Typickou strukturu obvodu FPGA znázorňuje následující obrázek 3.1., kde je názorně vidět maticová struktura logických bloků a propojovacích vodičů. Zde označené bloky IOB (Input/Output Block) jsou vstupně-výstupní obvody pro odpovídající vstupně-výstupní piny obvodu FPGA. Dále jsou uvedeny bloky CLB (Configurable Logic Block) představující programovatelné logické bloky (viz obr. 3.2.). Pro realizaci dané úlohy je pak použito konkrétní propojení těchto bloků pomocí globálních propojovacích matic PSM (Programmable Switch Matrix).



Obr. 3.1. Struktura propojení prvků FPGA obvodu

Obvod FPGA, který je v diplomové práci použit, patří mezi výrobky firmy Xilinx. Konkrétně se jedná o typovou řadu Spartan-6 [8], která je založena na technologii 45nm výrobního procesu, a vyznačuje se dobrým poměrem ceny a výkonu. Výrobce nabízí celkem 13 druhů čipů této řady, které disponují 3840 až 147433 logickými bloky a jsou osazeny od 132 až po 540 uživatelských vstupně výstupních pinů.



Obr. 3.2. Blokové schéma uspořádání CLB bloku [8]

Každý logický blok v obvodu Spartan-6 obsahuje dva obvody „Slice“, které se dále rozlišují na tři typy a to SLICEX, SLICEL a SLICEM. Jejich zastoupení v obvodu je 50% SLICEX a po 25% SLICEL a SLICEM. Každý obvod „Slice“ obsahuje 4 bloky LUTs (Look-Up Tables) se 6 adresovatelnými vstupy a 8 klopných obvodů. V tabulce 2. jsou popsány dostupné prvky k jednotlivým obvodům.

Pro obvod FPGA Spartan-6 XC6SLX45, který je osazen na vývojové desce Atlys (viz dále) jsou tyto další technické specifikace: 43661 logických buněk, které dohromady obsahují 54576 klopných obvodů a 401kB distribuované paměti RAM. Pouzdro čipu CSG324, které disponuje 218 uživatelskými piny. Pro každý uživatelský vstupně výstupní pin je možné přiřadit určitý standard až do napětí 3,3V, včetně standardu pro TMDS protokol.

Tabulka 2. Prvky obvodu Spartan-6 [8]

PRVKY OBVODU „SLICE“	SLICEX	SLICEL	SLICEM
6-vstupový LUTs	•	•	•
8 klopných obvodů	•	•	•
Multiplexor		•	•
Carry logic		•	•
Distribuovaná paměť RAM			•
Posuvný registr			•

3.2 Vývojová deska Digilent Atlys

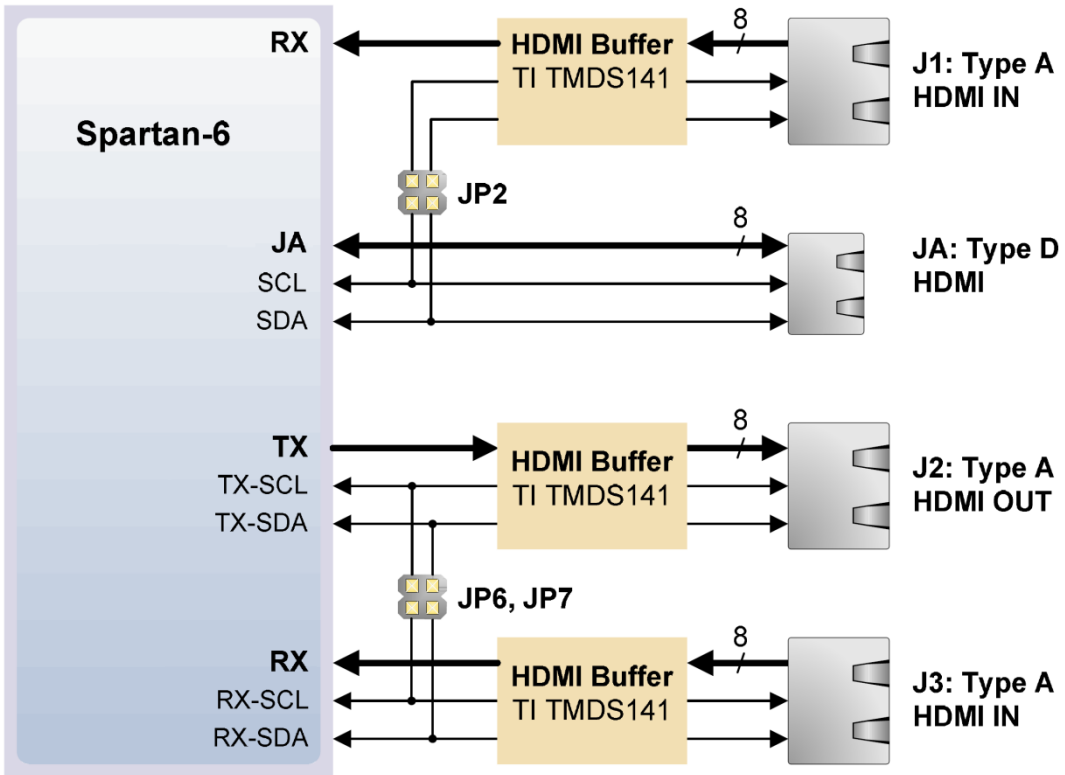
Vývojová deska Digilent Atlys [2] je komplexním řešením pro návrh různých zařízení, resp. projektů pro aplikace na FPGA obvodu Spartan-6. Deska je vedle hradlového pole (čipu) osazena sadou moderních periferií, které zajišťují jak komunikaci s okolím, např. GbitEthernet, HDMI konektory, audio a USB konektory, tak i podporu pro práci s daty, např. paměť DDR2, paměť flash. Díky těmto vlastnostem je vývojová deska Atlys vhodným zařízením pro vytvoření vestavěných systémů na bázi FPGA obvodu, které můžou být řízeny například softwarovým procesorem MicroBlaze. Deska je plně kompatibilní se všemi nástroji firmy Xilinx, včetně zdarma dostupného vývojového nástroje WebPack, díky čemuž je možné vytvořit celý systém bez dalších nákladů.

Důležité parametry pro práci s vývojovou deskou Atlys:

- FPGA obvod Xilinx Spartan-6 LX45
- 128MB DDR2 paměť
- 10/100/1000 Ethernet rozhraní
- USB port pro programování FPGA obvodu
- USB-UART a USB-HID port
- 4 HDMI porty

- zvukový kodek AC-97 s konektory line-in, line-out, microphone a headphone
- 16MB Flash paměť
- 100MHz oscilátor
- Rozšiřující konektor s 48 vstupně-výstupními piny
- 8 indikačních LED diod, 6 tlačítek a 8 přepínačů

Deska obsahuje čtyři HDMI konektory (viz obr. 3.3.), jedná se o tři standardní konektory typu A, které jsou nastaveny jako dva vstupní (J1,J3) a jeden výstupní konektor (J2). Tyto konektory jsou navíc opatřeny čipem TMDS141 firmy Texas Instruments, který slouží jako zesilovač signálu. Poslední čtvrtý konektor (JA) je na desce přítomen ve formě konektoru typu D, jedná se o tzv. mikro konektor o rozměrech 2.8 mm × 6.4 mm, který se nachází na spodní straně vývojové desky. Tento konektor již není opatřen výše zmíněným čipem TMDS141 a je připojen přímo na (vstupní/výstupní) piny obvodu FPGA. Výhodou je možnost použít konektor buď jako vstupní nebo jako výstupní. Na druhou stranu v důsledku toho nabízí menší přenosové rychlosti nebo nutnost použit kratší propojovací kabel – jinak je možné rušení signálu nebo slabý signál.



Obr. 3.3. Uspořádání komunikačních rozhraní HDMI na desce Atlys [2]

Jelikož je deska opatřena pouze konektory HDMI, které nepodporují přenos analogového signálu, je možné přijímat a vysílat pouze digitální obrazový signál (monitory s pouze analogovým vstupem není možné řídit). Z 19 pinových HDMI konektorů nejsou k FPGA připojeny všechny signály. Z každého konektoru jsou připojeny pouze 4 diferenciální signály pro přenos dat (na obr. 3.3. tlustá čára) a signály I2C sběrnice (na obr. 3.3. dvojice slabých čar). Jelikož většina zdrojů videosignálu potřebuje pro zahájení vysílání dat dostat informaci z I2C kanálu o připojeném zobrazovacím zařízení, jsou na desce přítomny propojky (na obr. 3.3. označeny JP2, JP6 a JP7), pomocí kterých je možné tuto informaci zprostředkovat přímo mezi zdrojem signálu a zobrazovacím zařízením, která jsou připojená k desce.

Nastavení (programování) FPGA obvodu je buď pomocí PC přes JTAG (Joint Test Action Group) rozhraní, které je na desce přítomno jak v podobě USB konektoru, tak v podobě konektoru pro plochý kabel v konfiguraci 2x7pinů, nebo je možné použít flash paměť na desce nebo je možné použít USB flash disk připojený do USB HID portu. Poslední dvě možnosti kompenzují nevýhodu oproti např. CPLD (Complex Programmable Logic Device) obvodům, jelikož vnitřní propojení FPGA obvodu se po přerušení napájení vymaže.

3.3 Jazyk VHDL

K popisu hardwarových struktur nejen obvodů FPGA se v dnešní době využívá programovacích jazyků typu HDL (Hardware Description Langure). Příkladem nejčastěji používaných jazyků tohoto typu může být například jazyk Verilog nebo jazyk VHDL (Very High Speed Integrated Circuits HDL) [5], [6], pomocí kterého byla napsána většina praktické části diplomové práce.

Jazyk VHDL byl původně vyvinut pro účely dokumentace a popisu číslicových systémů. Následně byl využíván pro simulaci těchto obvodů a později i k jejich logické syntéze. Za tímto účelem byl v roce 1987 poprvé definován standardem IEEE 1076. Do dnešní doby prošel jazyk několika revizemi, poslední v roce 2008, ovšem nejpoužívanější verzí je revize z roku 1993 (IEEE 1076-1993), kterou používám i v mojí práci.

Použití jazyka VHDL je výhodné pro jeho univerzálnost, neboť implementace pro navrženou strukturu je závislá až na výsledné syntéze VHDL kódu, což znamená, že s pomocí uvedeného jazyka můžeme provádět konkrétní návrhy pro hradlová pole různých výrobců (Xiling, Lattice, Altera ap.) a pak lze následně využít vhodný program pro logickou syntézu, přičemž tento program je ve většině případů dostupný ve volně šiřitelné verzi.

3.4 Návrhové prostředí Xilinx ISE Design

Jedná se o kompletní balík vývojových nástrojů pro hradlová pole výrobce Xilinx, který obsahuje nástroje od volně dostupného ISE WebPack až po kompletní nástroje pro vývoj vestavěných systémů nebo návrhů založených na digitálních signálových procesorech a to včetně nejrůznějších IP jader.

V rámci diplomové práce byly použity následující nástroje:

Project Navigator, ve kterém je možné navrhovat logické obvody za použití jazyka VHDL/Verilog nebo například schematických symbolů, a dále tyto návrhy pomocí externího nástroje ISim detailně simulovat.

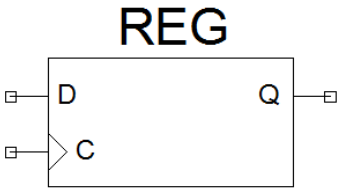
EDK (Embedded Development Kit) – jedná se o 2 programy XPS (Xilinx Platform Studio) a SDK (Software Development Kit) pro vývoj vestavěných systémů. XPS – slouží k vývoji hardwarových platforem pro vestavěné systémy, založené na softwarovém procesoru MicroBlaze nebo PowerPC. Umožňuje rychlé sestavení systému za pomocí nejrůznějších průvodců. SDK – slouží k napsání softwaru, který bude ovládat vestavěný systém. Umožňuje vývojářům psát, kompilovat a ladit C/C++ aplikace pro jejich vestavěné systémy s možností odzkoušení programu v simulátoru nebo přímo na připojeném FPGA obvodu (vývojových deskách).

Core Generator - slouží ke generování parametrizovaných IP jader (Intellectual Property cores). Tyto jádra mají výhodu, že jsou optimalizovaná pro použití na hradlových polích firmy Xilinx. Nástroj přináší celou řadu IP jader od základních, jako např. fronty nebo různé druhy pamětí až po složitá a komplexní jádra, jako např. různé sběrnice (např. PCI nebo PCI-X) nebo různé komunikační a síťová rozhraní (např. Ethernet).

4 Návrh a realizace video efektů pomocí FPGA

V této kapitole jsou popsány vlastní algoritmy a jejich hardwarová realizace pro řešení úloh barevného zpracování signálu (kapitola 4.1) a pro aplikaci konvolučního součtu video signálu s konvoluční maticí (maskou) o rozměru 5x5 (kapitola 4.2). Jednotlivé algoritmy byly zpracovány v jazyce VHDL. Zdrojový kód programu je v diplomové práci připojen na CD-ROMu, zde jsou pro přehlednost algoritmy popsány s podporou blokových schémat, přičemž u vícebitových signálů je dle zvyklostí uveden v závorce jejich rozsah.

Na začátku této kapitoly uvedu ještě popis bloku, který je využit v dalších schématech obou podkapitol. Jedná se sice o algoritmicky jednoduchý blok, který pomocí klopných obvodů D, realizujících jednobitovou paměť, umožňuje zpoždění vstupního signálu po dobu jedné periody hodinového signálu. Použití tohoto bloku při návrhu algoritmů pro FPGA obvody je důležité pro snížení kombinačního zpoždění na kritických cestách, což vede k zvýšení frekvence, na které bude obvod pracovat. Označení pro zakreslení v následujících blokových schématech je na obr. 4.1., kde D je označení vstupního a Q výstupního signálu a C je hodinový signál. Pro další zjednodušení schémat je sériové řazení tohoto bloku označeno jejich počtem, tj. např. pro tři bloky typu REG řazené za sebou je použit jeden blok s označením REG-3.

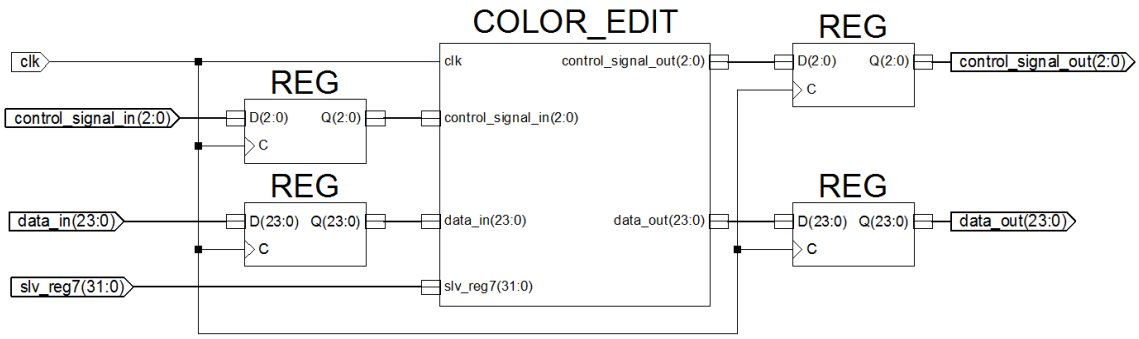


Obr. 4.1. Blok jednobitové paměti

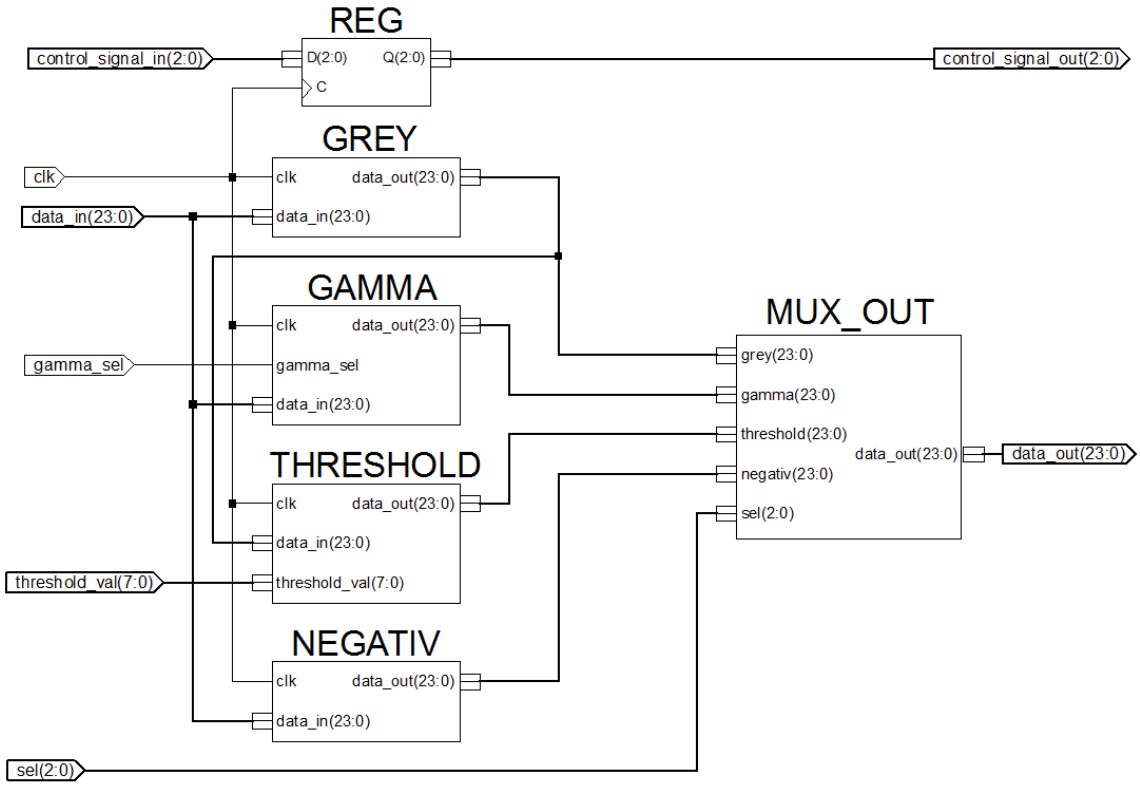
4.1 Implementace základních úprav barev

Všechny úpravy barev jsou prováděny v jednom samostatném bloku. Tento blok obsahuje naprogramované algoritmy vytvoření černobílého obrazu, gama korekci,

prahování a vytvoření negativu. Vstupem tohoto bloku (viz obr. 4.2) je hodinový signál **clk**, tři prvky řídicího signálu **control_signal_in** sloužící ke korektnímu zobrazení videosignálu na monitoru, 24bitová video data **data_in** a jeden registr **slv_reg7** o délce 32 bitů, který slouží k přenosu dat, která jsou uživatelsky nastavitelná. Na výstupu jsou upravená video data **data_out** a k nim patřičně zpožděné (časově synchronizované) řídicí signály **control_signal_out**.



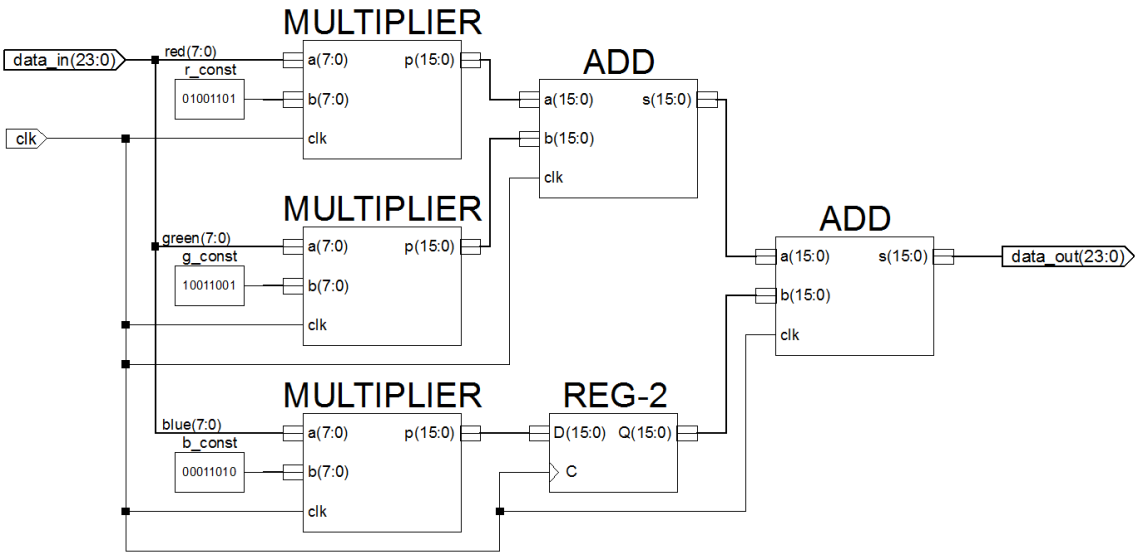
Obr. 4.2. Schéma připojení vstupních a výstupních dat bloku pro úpravu barev



Obr. 4.3. Schéma vnitřního uspořádání bloku pro úpravu barev

Vnitřní uspořádání bloku COLOR_EDIT je znázorněno na obrázku 4.3. Je zde znázorněno zapojení jednotlivých bloků realizující výše zmíněné úpravy barev, včetně propojení s výstupním multiplexorem, který zajišťuje přepínání mezi výstupy z jednotlivých bloků pro úpravu barev a konečným výstupem z bloku. Tento multiplexor je řízen uživatelsky nastavitelným 3bitovým vstupem **sel**, který je součástí vstupního signálu **slv_reg7** (viz obr. 4.2), stejně tak i další řídicí signály **gamma_sel** pro algoritmus gama korekce a 8bitový **threshold_val** pro funkci prahování.

Schéma vnitřního uspořádání bloku GREY pro úpravu šedotónového obrazu je na obr. 4.4. Jak již bylo zmíněno v kapitole 2.1.1., vytvoření tohoto obrazu z barevné předlohy není vzhledem k lidskému oku ideální pomocí obyčejného průměrování barev RGB, ale používá se vážený průměr s hodnotami 0.299 pro červený kanál, 0.587 pro zelený kanál a 0.114 pro modrý kanál. Toho bylo docíleno vynásobením jednotlivých konstant hodnotou 256 a poté následným zaokrouhlením, čímž bylo získáno celé číslo, kterým již lze jednoduše v hradlových polích násobit. Vynásobením jednotlivých barevných složek pixelu příslušnými konstantami a jejich sečtením byla získána informace o jasu pixelu. Následně bylo ještě potřeba provést dělení hodnotou 256, což se provedlo bitovým posunem čísla ve dvojkové soustavě o 8 pozic vpravo.

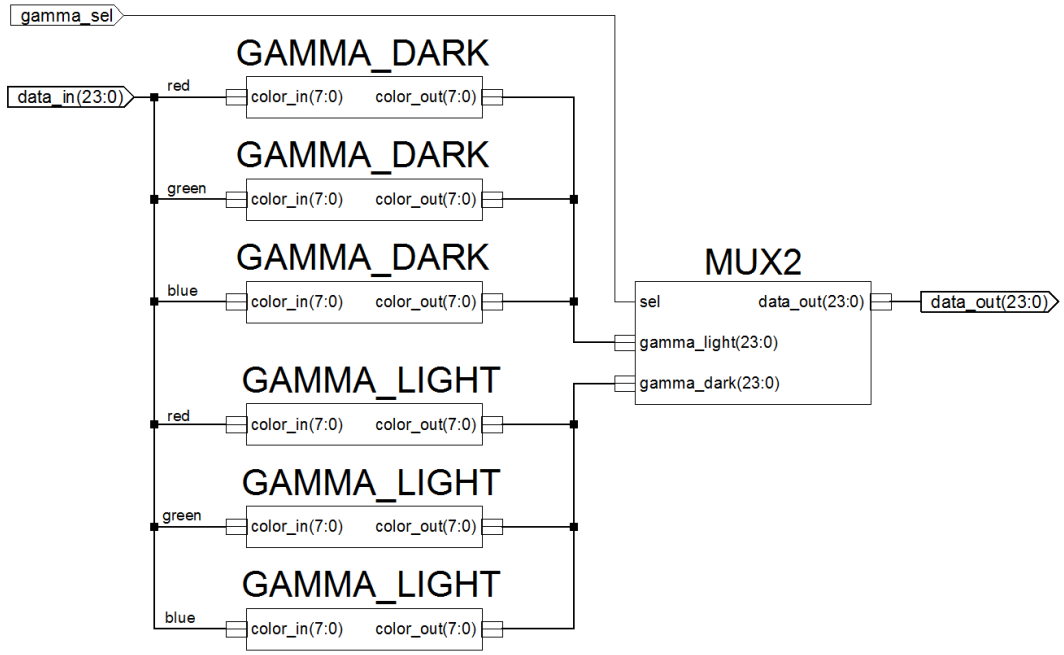


Obr. 4.4. Schéma vnitřního uspořádání bloku pro úpravu šedotónového obrazu

K aplikaci gama korekce jsem pro realizaci výrazu na pravé straně rovnice (2) použil z praktických důvodů předem vypočtených (navzorkovaných) dat pro celočíselné hodnoty intenzity i z intervalu <0, 255>. Vznikl tak vektor tabelovaných hodnot, kde pořadové číslo prvku vektoru je hodnota intenzity i a obsah tohoto prvku udává odpovídající hodnotu intenzity i_1 podle upraveného vztahu (7).

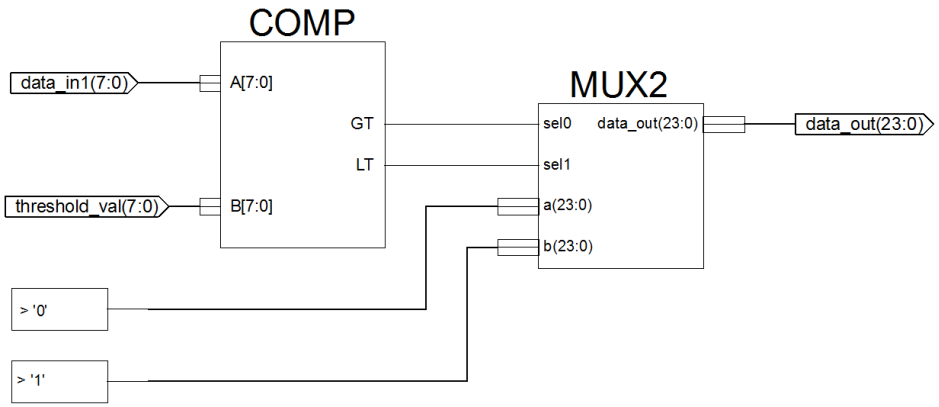
$$i_1 = \left[\left(\frac{i}{255}\right)^{\frac{1}{\gamma}}\right] \cdot 255 \qquad (7)$$

Pro blok GAMMA byly takto navrženy dvě korekce vstupního obrazu, a to pro zesvětlení hodnota $\gamma = 1{,}8$, pro ztmavení $\gamma = 0{,}5$. Tyto hodnoty je však možné podle požadavku změnit. Pro uložení navzorkovaných dat byla místo paměti RAM z praktických důvodů zvolena paměť ROM, která sice neumožní operativně přepsat podle potřeby navzorkované hodnoty, ale podstatně sníží nároky na použitou paměť. Pro každou navzorkovanou sadu hodnot byla navržena paměť o velikosti 2Kb (GAMMA_DARK a GAMMA_LIGHT) - viz obr. 4.5. Každý tento blok paměti ROM byl rozdělen na 256 buněk, každé o velikosti 8 bitů. Adresy buněk (color_in) odpovídají vstupní hodnotě barvy, výsledná hodnota barvy (color_out) je pak dána obsahem této buňky.



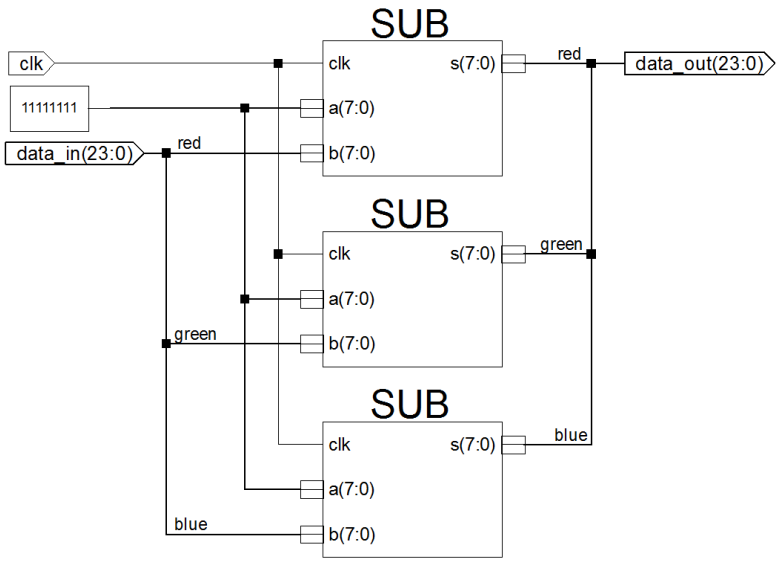
Obr. 4.5. Schéma vnitřního uspořádání bloku pro gama korekci

Dalším navrženým blokem je THRESHOLD. K výpočtu prahování se zde používá komparátoru, který porovnává jasovou hodnotu aktuálního pixelu (**data_in**) s prahovou hodnotou **threshold_val** (obr. 4.6), která je uživatelsky nastavitelná v rozsahu 0 až 255.



Obr. 4.6. Schéma vnitřního uspořádání bloku pro výpočet prahování

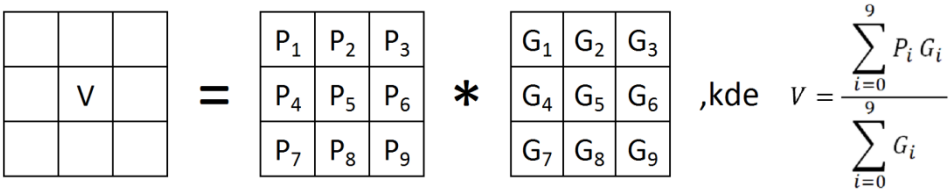
Posledním navrženým blokem je NEGATIV. K vytvoření negativu stačí pro jednotlivé barevné kanály vytvořit jejich doplněk do hodnoty 255, tj. pro každý barevný kanál je od hodnoty 255 odečtena aktuální hodnota barvy pixelu. Navržené zapojení je na obr. 4.7., kde potřebný doplněk pro každou barvu zvlášť realizuje blok SUB.



Obr. 4.7. Schéma vnitřního uspořádání bloku pro vytvoření negativu

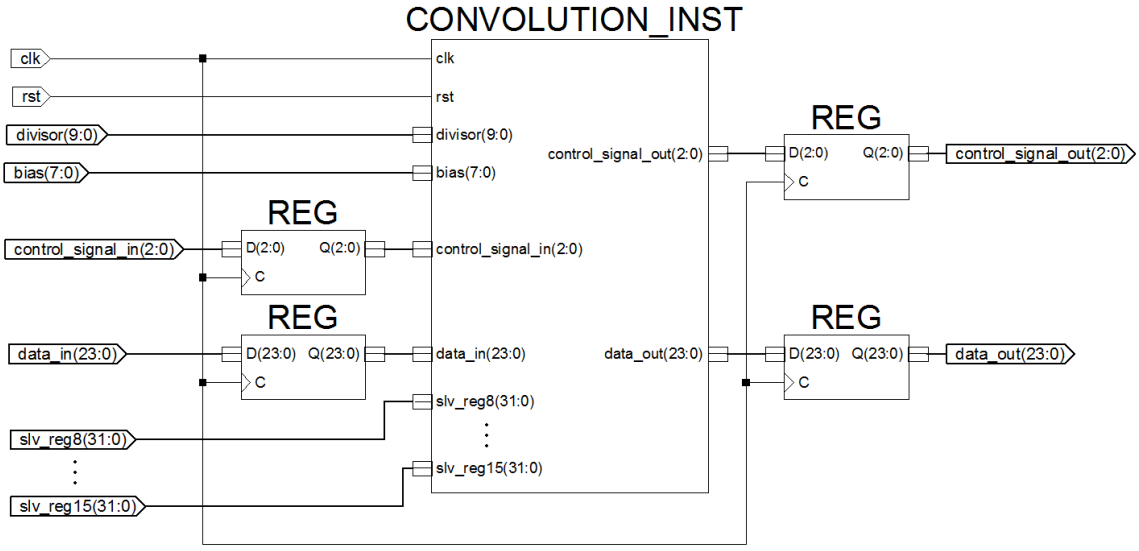
4.2 Implementace konvolučního filtru

Význam aplikace konvolučního filtru byl popsán v kapitole 2.2.1. V další části je popsán algoritmus realizující rovnici (6) konvolučního výpočtu. Pro názornost můžeme navrženou strukturu výpočtu tohoto algoritmu pro FPGA obvod demonstrovat na příkladě konvoluční matice o rozměru 3x3 dle schématu na obr. 4.8., kde "maticový výřez" obrazových dat je značen symboly P_1-P_9, prvky konvoluční matice jsou značeny symboly G_1-G_9 a V je výsledná hodnota nového obrazového bodu.



Obr. 4.8. Schéma výpočtu konvoluce pro konvoluční matici rozměru 3x3

Základní propojení vstupních a výstupních signálů s algoritmem konvolučního filtru je uvedeno na obr. 4.9. Konvoluční filtr a všechny jeho součásti potřebné k jeho správné funkci jsou zahrnuty v bloku CONVOLUTION_INST. Pro realizaci v rámci této práce byl zvolen rozměr konvoluční matice 5x5.



Obr. 4.9. Schéma připojení vstupních a výstupních dat výpočtu konvolučního filtru

Vstupními signály bloku CONVOLUTION_INST jsou:

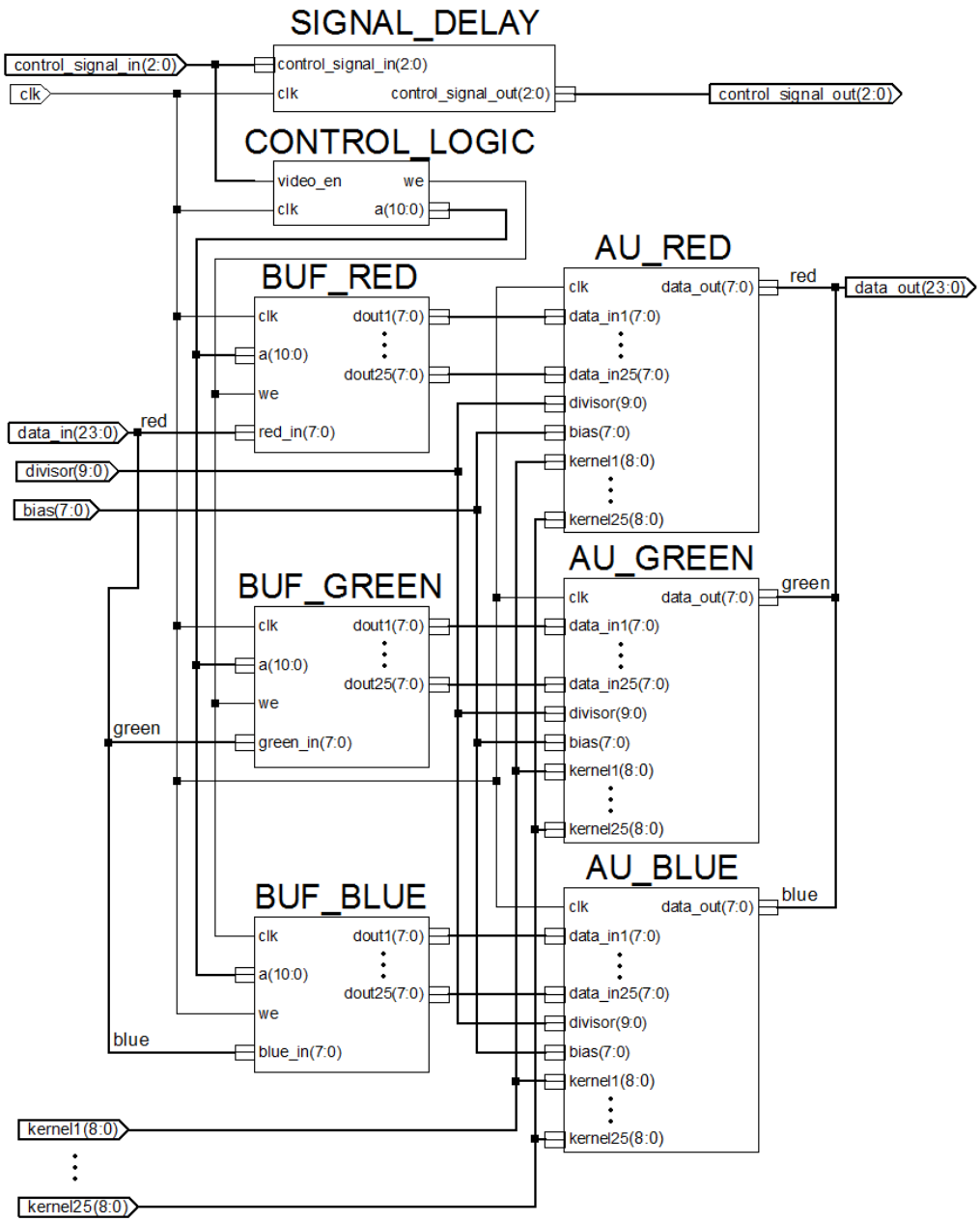
- Hodinový signál **clk**, který má stejnou frekvenci, jako výstupní hodinový signál ze zdroje videa, tzv. pixel clock.
- Signál **res** pro nastavení inicializačních hodnot (resetování).
- Signály **divisor** a **bias** pro výpočet konvolučního filtru. Hodnota signálu divisor je dána hodnotou jmenovatele výrazu pro výpočet hodnoty V dle vtahu na obr. 4.8. Samostatné zadání hodnoty divisor umožní uživateli ovlivnit jas upraveného obrazu. Hodnota signálu bias je prostým přičtením další možná uživatelem zvolená korekce hodnoty V.
- 3bitový signál **control_signal_in** obsahující jednak řídicí signál video_en, jenž určuje čas, po který jsou zdrojem videa vysílána viditelná obrazová data a dále dva řídící signály v_sync a h_sync pro vertikální a horizontální synchronizaci. Tyto dva signály jsou do bloku konvolučního filtru zapojeny pouze z důvodu jejich potřebného zpoždění tak, aby byly správně synchronizovány s vypočítanými obrazovými daty, která jsou v průběhu výpočtů zpožďována.
- Obrazová data ve formátu RGB přiváděná 24bitovým signálem **data_in**.
- Osm uživatelských registrů **slv_reg8** až **slv_reg15** o celkové kapacitě 256 bitů, které slouží k uživatelským změnám, jako například k nastavení koeficientů konvoluční matice.

Na straně výstupu jsou pak následující signály:

- 3bitový signál **control_signal_out** se signály video_en, v_sync a h_sync, které prošly požadovaným zpožděním.
- 24bitový signál **data_out** upravených obrazových dat.

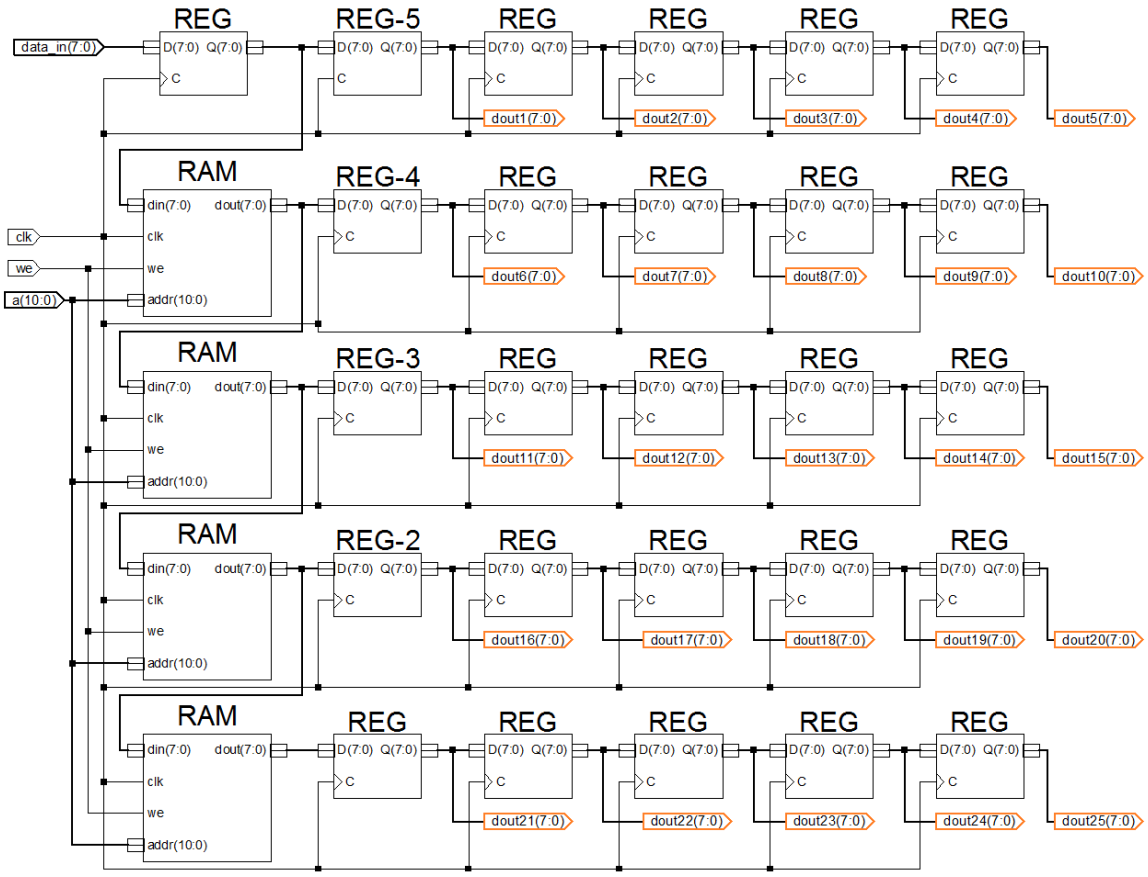
Schéma zapojení uvnitř bloku CONVOLUTION_INST je znázorněno na obr. 4.10. Zde jsou jednak dva bloky řídící logiky a jednak bloky, které realizují výpočet algoritmu konvolučního filtru.

Blok SIGNAL_DELAY zajistí pomocí posuvného 3bitového registru zpoždění tří již zmíněných řídících signálů (video_en, v_sync a h_sync). Doba jejich zpoždění je určena zpožděním, které vzniklo v důsledku průchodu obrazových dat **data_in** výpočetní logikou. Toho je docíleno nastavením délky posuvného registru, která odpovídá počtu period hodinového signálu mezi vstupními obrazovými daty a příslušnými výstupními obrazovými daty.



Obr. 4.10. Základní blokové schéma výpočtu konvolučního filtru

Vlastní realizaci konvolučního součinu (resp. násobení dvou matic velikosti 5x5 prvků) zajišťují tři bloky AU_RED, AU_GREEN a AU_BLUE. Data pro jednotlivé bloky jsou pak připravena v dalších třech blocích BUF_RED, BUF_GREEN a BUF_BLUE. Bloky s prefixem „BUF“ v názvu plní funkci vyrovnávací paměti, která má za úkol zpřístupnit data výpočetní jednotce. Struktura všech tří bloků je shodná a je znázorněna na obr. 4.11.



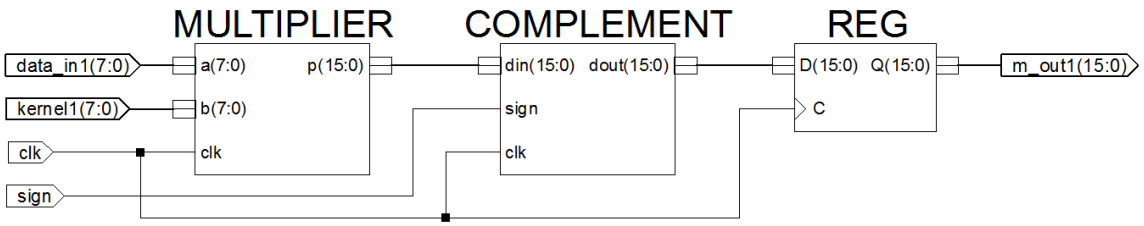
Obr. 4.11. Blokové schéma výběru obrazových dat pro výpočet konvolučního filtru

Každá barevná složka obrazového signálu tak má svoji část vyrovnávací paměti (bloky RAM), která uchovává čtyři řádky příslušné barvy z obrazového signálu a umožňuje tak za pomocí pátého, aktuálně načítaného řádku, vytvořit matici 5x5. Obrazová data jsou načítána přes vstupní registr, ze kterého jsou posílána jednak do posuvného registru, který reprezentuje první řádek konvoluční matice a jednak do paměti RAM, která slouží jako vyrovnávací paměť pro druhý řádek. Z této paměti jsou data posílána do dalšího posuvného registru, tentokrát reprezentující druhý řádek

konvoluční matice a do paměti RAM pro uložení třetího řádku. Tento algoritmus se pak opakuje stejně pro čtvrtý a pátý řádek. Velikost paměti RAM je 2048bitů a její šířka je 8bitů, umožňuje tudíž uložit jednu barvu z řádku o maximální délce 2048 pixelů.

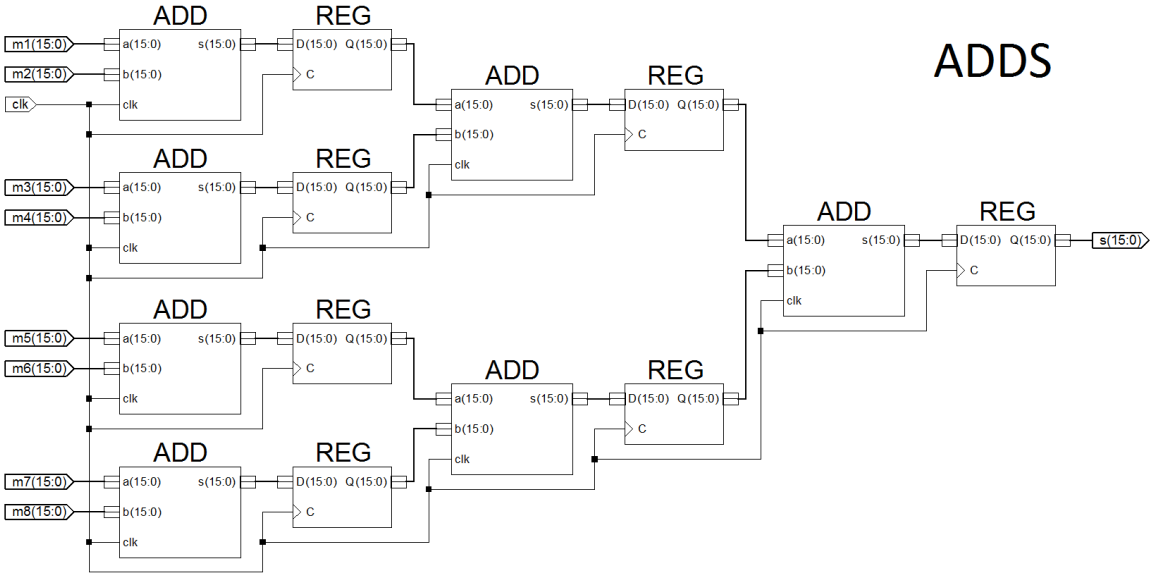
Mezi vstupy tohoto bloku je i vstup signálu „Write Enable“, označen jako **we**, který nastavuje povolení zápisu do paměti RAM pouze v době, kdy jsou přijímána viditelná obrazová data, což má za následek požadavek na nižší velikost paměti RAM, než kdyby se ukládala i data, která nejsou ve viditelné oblasti. Dalším vstupem je 10bitový signál **addr**, obsahující adresu, na kterou jsou data v paměti RAM ukládána, resp. čtena. Oba tyto signály jsou generovány pomocí řídící logiky v nadřazeném bloku CONTROL_LOGIC, viz schéma na obr. 4.10. Generování signálu **addr** je zajištěno pomocí inkrementálního čítače, který čítá pouze v době, kdy je aktivní výše zmíněný signál **we** a na konci každého čítaní je jeho hodnota resetována. Výstupem je 25 8bitových signálů **dout1** až **dout25**, které jsou brány z posuvných registrů a která tvoří matici obrazových dat pro konvoluční součin, který je pak realizován opět po jednotlivých barvách signálu RGB v blocích s prefixy AU.

Bloky označené prefixem AU tedy vypočítají nová obrazová data na základě struktury výstupních obrazových dat z bloku BUF a struktury vlastní konvoluční matice, jejíž prvky jsou dány obsahem 8bitových signálů **kernel1** až **kernel25.** Tyto bloky s označením AU tak představují aritmetickou jednotku konvolučního filtru. Hlavním stavebním prvkem tohoto bloku jsou násobičky a sčítačky. Vstupní data jsou nejdříve paralelně vynásobena pomocí 25 8bitových násobiček (viz schéma na obr. 4.12.), přičemž s ohledem na znaménko koeficientu je eventuelně proveden dvojkový doplněk v bloku COMPLEMENT.



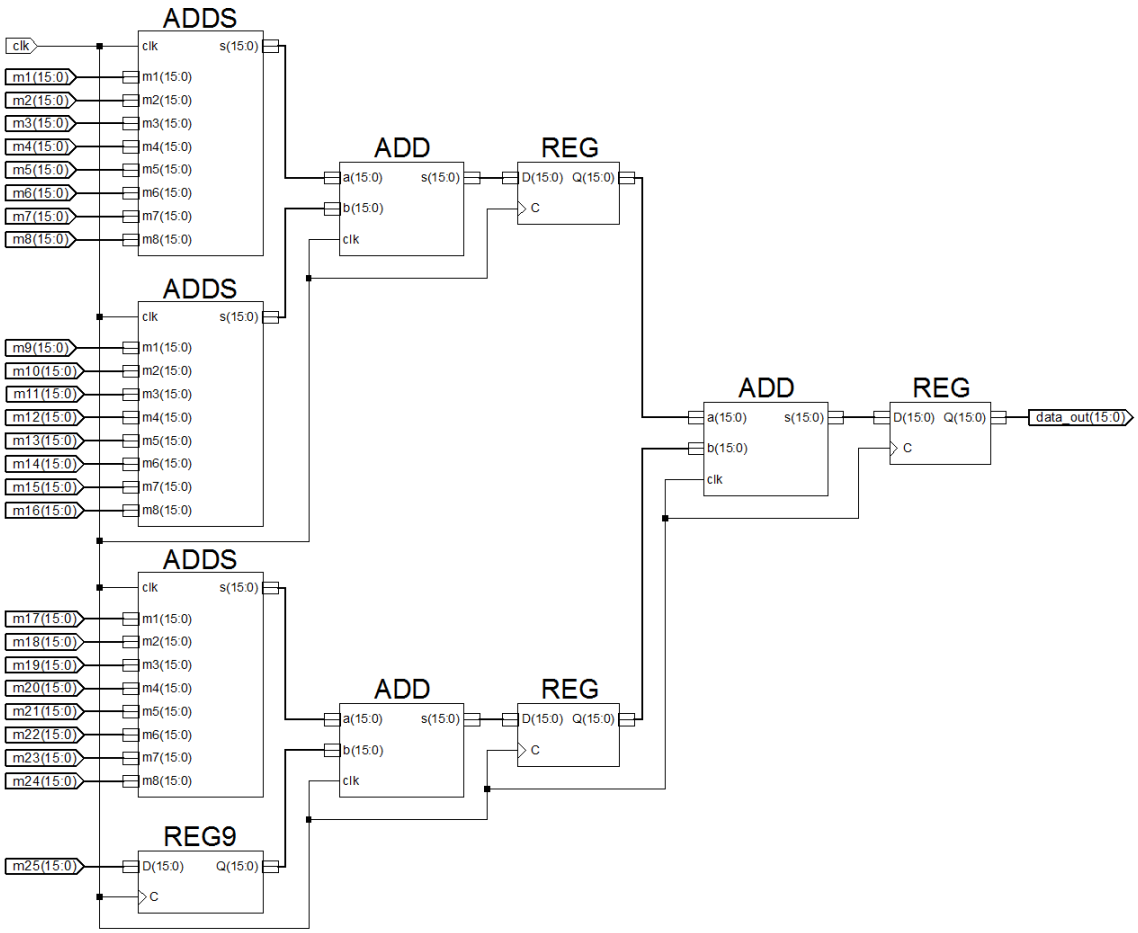
Obr. 4.12. Blokové schéma pro část součinu dat při výpočtu konvolučního filtru

Následně je za pomoci 24 součtových bloků (vždy pro 2 signály) provedeno postupné sčítaní všech součinů. Pro znázornění této části je algoritmus rozdělen do dvou schémat. První schéma na obr. 4.13. jen ilustrativně znázorňuje postupné sečtení osmi signálů a na obr. 4.14. je pak znázorněno propojení pro součet všech 25 signálů (výsledky násobení odpovídajících prvků pro součin dvou matici o rozměru 5x5). Blok ADDS zde reprezentuje aplikaci schéma z obr. 4.13.



Obr. 4.13. Blokové schéma pro část výpočtu konvolučního filtru

Hodnota tohoto konvolučního součtu je následně upravena na základě hodnot signálů **divisor** a **bias** a poté převedena pomocí multiplexoru na konečný výsledek, který bude reprezentovat bílou barvu v případě, že byl výsledek sčítání větší než hodnota 255, černou barvu v případě, že byl výsledek sčítání záporný a v ostatních případech zůstane zachována původní hodnota výpočtu. Výstup z multiplexoru je následně poslán na výstup z bloku CONVOLUTION_INST jako výstup celého algoritmu.



Obr. 4.14. Blokové schéma pro sumační část výpočtu konvolučního filtru

4.3 Sestavení vestavěného systému

Poslední částí vlastní realizace navržených algoritmů bylo jejich odladění na zkušebních datech a vytvoření funkčního modulu, který je pak použitelný jako tzv. vestavěný systém.

Pro ověření správných výpočtů sestavených algoritmů dle kapitol 4.1. a 4.2. jsem využil v rámci programovacího jazyka VHDL standardně dostupných prostředků pro simulaci navržených zapojení hradlových polí. Pomocí těchto prostředků je možné sestavit „konfigurační“ soubor simulace, kde lze definovat a měnit hodnoty jednotlivých vstupních signálů příslušné entity a to v čase po jednotlivých periodách hodinového

signálu. Dále je možné nastavit parametry simulace jako např. délku periody hodinového signálu a jeho střídu a také délku celé simulace.

V simulaci je možné sledovat průběh libovolných signálů, které se v návrhu vyskytují. Po prvním spuštění jsou standardně zobrazeny pouze vstupní a výstupní signály příslušné simulované entity a uživatel může dále přidávat další jednotlivé signály ze seznamu nebo naopak jednotlivé signály odebírat pro větší přehlednost.

Simulační výpočty jsou v podstatě nezbytné pro ověření správného sestavení výsledné realizace. Pomocí testovacích dat se dají odladit případné chyby jak v zapojení (interpretaci) blokových schémat, tak i případně ve vlastním navržení algoritmů. K tomu je však potřeba mít dobře připravená vstupní data tak, aby výsledky simulací byly dostatečně věrohodně ověřitelné. Proto jsem nejprve připravil sadu účelově vymyšlených dat, na kterých jsem byl schopen výpočetně (ručně nebo s využitím Excelu) ověřit výsledky jednotlivých algoritmů dle schémat v předchozích dvou kapitolách. Nejdříve jsem simuloval jednoduchá data pro ověření správného časování systému. Po odladění časování jsem simuloval vybraná data (výřez) skutečného snímku reálných dat. Simulační výpočty mi opravdu pomohly některé (někdy téměř skryté) chyby odhalit. Řádně odsimulovaný systém je pak již připraven k vyzkoušení v reálné aplikaci.

K sestavení a oživení celého aplikačně připraveného systému jsem využil k tomu účelu určený softwarový balík Xilinx EDK, který obsahuje integrované vývojové prostředí Xilinx Platform Studio včetně IP jader a vývojové prostředí Software Development Kit, které slouží pro napsání řídícího programu procesoru. V rámci těchto prostředků byly pro jednotlivé návrhy algoritmů ve VHDL vytvořeny příslušné moduly (jádra) a připojeny další komponenty, které slouží ke komunikaci FPGA obvodu s okolím, jako např. vstup hodinového signálu z oscilátoru, vstupy a výstupy z HDMI konektorů nebo třeba vstupy z uživatelských přepínačů na vývojové desce Atlys. Pro zajištění kódování a dekódování dat podle TMDS protokolu jsem použil komponentu, která je volně k dispozici od firmy Xilinx. Celý systém je řízen 32bitovým softwarovým procesorem MicroBlaze a ke komunikaci s jednotlivými moduly je použita sběrnice AXI (Advanced eXtensible Interface). Následně byl vygenerování tzv. bitstream

souboru, pomocí kterého se obvod FPGA konfiguruje. Posledním krokem pro funkčnost celého systému je napsání řídícího programu pro procesor MicroBlaze.

Popsané a navržené řešení zpracování video signálu je pro konečnou realizaci koncipováno jako samostatný modul, který je možné připojit mezi zdroj signálu a zobrazovací zařízení, jak je znázorněno na obr. 4.15.

Obr. 4.15. Zapojení výsledného modulu

5 Demonstrační úloha

Poslední kapitola je věnována praktickým ukázkám výstupu jednotlivých algoritmů pro navržené zpracování video efektů. Jako příklad do této kapitoly byl vybrán jeden náhodný snímek snad s jedinou přirozenou podmínkou, aby byl dostatečně barevně bohatý a v dobrém rozlišení. Dále byla zvolena hodnota prahování a hodnoty konvoluční matice pro detekci hran a zvýraznění hran. Tyto uživatelské hodnoty jsou zadávány do programu v souboru user_values.h dle následující struktury:

- Dvojice hodnot (sign01, kernel01) až (sign25, kernel25) pro zadání znaménka a absolutní hodnoty konvolučního jádra, přičemž kladnému znaménku odpovídá hodnota 0 a zápornému znaménku hodnota 1 a rozsah absolutních hodnot je dán z intervalu celých čísel <0, 255>.
- Hodnoty parametrů divisor a bias pro výpočet konvolučního filtru z intervalu <0, 1023>
- Hodnota prahování threshold_value v rozsahu celých čísel z intervalu <0, 255>.

Poté je program nahrán do FPGA obvodu a připraven ke spuštění úlohy.

Jako další řídicí prvky jsou využity ovládací prvky (přepínače) na desce Atlys. Pomocí nich lze nastavit vždy jen jednu úlohu video efektu pro zobrazení. Přiřazení aktivit jednotlivým tlačítkům je dle následující tabulky:

- 1 šedotónový obraz
- 2 gama korekce pro zesvětlení obrazu
- 3 gama korekce pro ztmavení obrazu
- 4 prahování
- 5 vytvoření negativu

Dále je možné tyto ovládací prvky na desce Atlys využít pro volbu konkrétní konvoluční matice z předem definovaného výběru, resp. modifikace programu

umožňuje zadat maximálně 6 konvolučních matic a pomocí tlačítek 1 až 6 je pak vybrána vždy jen jedna jako aktivní.

Pro zvolené demonstrační úlohy byla zvolena následující data:

Hodnoty konvoluční matice pro detekci hran a zvýraznění hran:

$$\begin{bmatrix} -1 & -1 & -1 & -1 & -1 \\ -1 & -2 & -2 & -2 & -1 \\ -1 & -2 & 32 & -2 & -1 \\ -1 & -2 & -2 & -2 & -1 \\ -1 & -1 & -1 & -1 & -1 \end{bmatrix} \qquad \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & -1 & 0 \\ 0 & -1 & 9 & -1 & 0 \\ 0 & -1 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Hodnota prahování threshold_value = 127

Hodnota gama korekce (nastavena implicitně) gamma = 1,8

Na závěr této kapitoly jsou uveden postupně jednotlivé výsledky:

Předloha barevného snímku (originál):

Úprava originálu na šedotónový obraz:

Úprava originálu gama korekcí:

Úprava originálu prahováním:

Úprava originálu vytvořením negativu:

Úprava originálu detekcí hran:

Úprava originálu zvýrazněním hran:

Závěr

Tématem diplomové práce je navržení a realizace vybraných úprav video signálů pomocí hradlových polí FPGA. K návrhu algoritmů a jejich simulačnímu ověření bylo využito vývojového prostředí Xilinx a programovacího jazyka VHDL. Pro vlastní technické řešení pak bylo použito vývojové desky Digilent Atlys s osazeným FPGA obvodem Spartan 6 XC6SLX45.

Výsledné řešení bylo odzkoušeno na video signálech z různých zdrojů včetně televizního vysílání. Pro dokumentaci výsledků pak byl vybrán jeden barevný snímek a v poslední kapitole jsou prezentovány výsledky jednotlivých úprav.

Navržené řešení je plně funkční a domnívám se, že je možné jej použít např. v domácích podmínkách pro "on-line" sledování, resp. úpravu videa, což bylo původně i dílčím cílem zadání. Dosažená rychlost zpracování, řádově milisekundy, však nabízí využití i všude tam, kde je potřeba v reálném čase vyhodnotit, případně nějak upravit snímaný videosignál pro další použití. Příkladem takového využití mohou být např. mobilní roboti, kteří mají rozpoznat a překonat překážky. Jejich uplatnění je pak samozřejmě velmi pestré, počínaje třeba jen sofistikovanými hračkami pro děti až po nasazení v záchranářském resp. vojenském prostředí.

Literatura

[1] DDWG: Digital Visual Interface DVI, 1999,
online <http://www.ddwg.org/lib/dvi_10.pdf>

[2] Digilent Inc.: Atlys Reference Manual,
online <http://www.digilentinc.com/Data/Products/ATLYS/Atlys_rm.pdf>

[3] Chaloupka, J.: Přednášky předmětu počítačové vidění,
online <http://visper.ite.tul.cz/vyuka/>

[4] Kaňka. M.: Implementace konvolučních filtrů na obvodu FPGA (semestrální), TUL 2009

[5] Marwedel, P.: Embedded system design, Springer 2006, ISBN 0-387-29237-3

[6] Pinker J., Poupa M.: Číslicové systémy a jazyk VHDL, BEN 2006,
ISBN 80-7300-198-5

[7] Skala V.: Světlo, barvy a barevné systémy v počítačové grafice, Academia 1993,
ISBN 80-200-0463-7

[8] Xilinx Inc.: Spartan-6 dokumentace,
online <http://www.xilinx.com/support/documentation/spartan-6.htm>

[9] Žára J., Beneš B. Sochor J., Felkel P.: Moderní počítačová grafika,
Computerpress 2004, ISBN 80-251-0454-0